(2)

Technical Document 2101
April 1991

Operating Systems Standards Working Group (OSSWG) Next Generation Computer Resources (NGCR) Program

First Annual Report— October 1990

R. Bergman/OSSWG

5

NAVAL OCEAN SYSTEMS CENTER
San Diego, California 92152-5000

J. D. FONTANA, CAPT, USN
Commander

H. R. TALKINGTON, Acting
Technical Director

ADMINISTRATIVE INFORMATION

The work reported here was conducted over the period January 1989 - September 1990 by a joint team of experts in the field of computer operating systems. These experts were from the Navy, other areas of government, industry, and academia. Only a few of the Navy participants were actually funded to directly participate in this process.

The report was funded under NOSC Job Order Number CC30410F01, Next Generation Computer Resources. The sponsoring activity is the Space and Naval Warfare Systems Command, through the work of the Operating Systems Standards Working Group (OSSWG). The OSSWG management structure was as follows for the performance period of January 1989 - September 1990:

NGCR Program Manager, Mr. H. Mendenhall, SPAWAR 324
NGCR OSSWG Co-Chairman, CDR R. Barbour, SPAWAR 324
NGCR OSSWG Co-Chairman, Ms. T. Oberndorf, NADC
Approach Subgroup Chairman, Mr. T. Conrad, NUSC
Requirements Subgroup Chairman, Mr. R. Bergman, NOSC
Available Technology Subgroup Chairman, Mr. J. Oblinger, NUSC

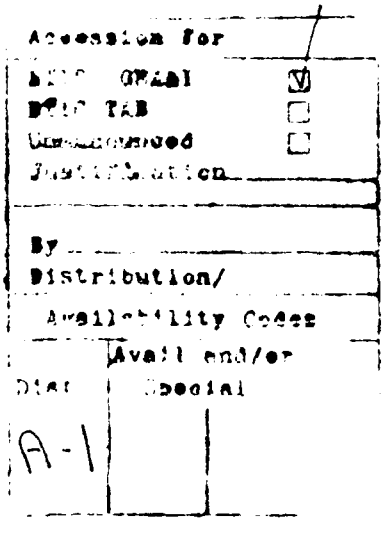
The products contained within this report are the result of work performed by the entire membership of the OSSWG.

Released by
L. J. Core, Head
Embedded Computer
Systems Branch

Under authority of
A. G. Justice, Head
Information Processing
and Displaying Division

RBT

Table of Contents



Part 1

Executive Summary

The Next Generation Computer Resources (NGCR) Program of the U.S. Navy seeks to establish standard interfaces of several types in order to provide an open system architecture for constructing Navy application systems from compatible components. These interfaces are to be based on industry / commercial standards. Among the standard interfaces sought is an operating system interface. The Operating Systems Standards Working Group (OSSWG) was formed in 1989 for the purpose of identifying such a standard. The OSSWG is open to participation by all interested parties and has met approximately once every six weeks in pursuit of its objectives. This report is the formal record of its achievements for the period January 1989 - September 1990. It is intended that an Annual Report will be issued each October to record the previous year's work.

Part 2 of this report documents the sequence of meetings held by the OSSWG and incorporates the formal minutes of each of those meetings. Excluded are the briefing materials used by the several subgroups and the invited briefs by various technical experts, although their message is reflected in the minutes.

Part 3 contains the principal products of the OSSWG for this reporting period. Included there is the original white paper on operating system interface standardization issues, based on which the OSSWG began its work. Also included is the Plan of Action and Milestones developed to guide the OSSWG's efforts. This POA&M addresses all the work accomplished through June 1990, a period now referred to as Phase 1 of the OSSWG's activity. During Phase 1, the work was shared by three OSSWG subgroups, each of which produced substantial results. The Approach Subgroup produced a Data Item Description for an Operational Concept Document for an operating system interface standard. Further, it generated a Reference Model for discussing operating system interfaces. Both of these are found in Part 3. The Available Technology Subgroup performed a survey of the operating system marketplace, as well as a survey of relevant standards, and produced an Operating System Technology Report documenting its findings. The Requirements Subgroup produced many drafts of a statement of Navy requirements for operating system interface capabilities, defining sixteen classes of requirements and appropriate metrics for each requirement. The end result was the Operating System Interface Requirements Document, Version 2.0, which is also included in Part 3 of this report.

Four other major reports were produced during Phase 1. These are not incorporated herein, but have been formally published and are available upon request by contacting the National Technical Information Service at the following address:

U.S. Department of Congress
National Technical Information Service
5285 Port Royal Road
Springfield VA 22161
(703) 487-4650.

The cover sheets from those reports as well as the Report Documentation Page and an executive summary are bound into this report. These four reports describe (1) the evaluation process defined for evaluating the candidate operating system interfaces and selecting a standard, (2) the results of applying that process to the finalist candidates, (3) the final recommendation of the OSSWG as to the selection of a standard, and (4) an "after action" analysis of what was learned and what should happen next.

In order to provide a suitable context for understanding the work reported here, a set of briefing charts describing the NGCR program in its entirety is appended below. Among the important facts revealed in the briefing charts is the method of work within the NGCR program. The essence of the approach is the joint industry/Navy working group. The progress documented in this Annual Report reflects the contributions of the members of the OSSWG, many of whom, especially those from industry and the academic community, participated on a voluntary basis.

Finally, those interested in the origination of the NGCR program can obtain further information in the following documents:

Operational Requirement for Next Generation Computers
CNOTRANSMITTAL 098r/8u55086 8 August 1988

Next Generation Computer Resources Development Options Paper
COMSPAWARSYSCOM 324/253 30 October 1987

NGCR POINT OF CONTACT LIST 12/11/90

To assist in maintaining a current NGCR program point of contact list, please fill in the following information for yourself and any other representatives from your organization deemed appropriate. Please print the information carefully and legibly. A complete list will be provided to each of you before your departure later in the week. Thank you for your cooperation.

1.) Name

2.) Organization and Code

3.) Telephone Number

4.) E-mail Address

5.) Fax Number

6.) Mailing Address

7.) Area of Program Involvement, e. g., S/NWG, PSEWG

U. S. NAVY

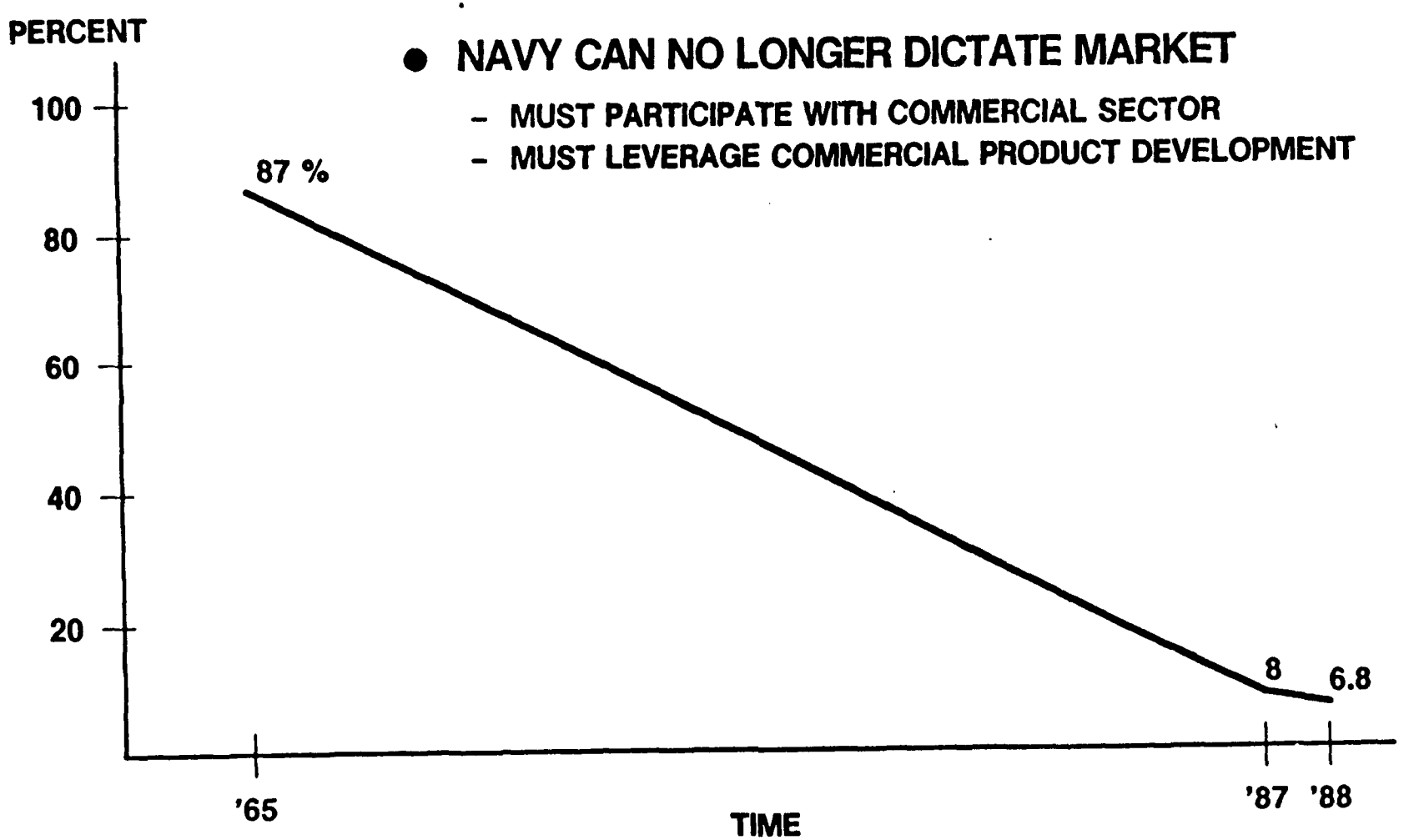
NEXT GENERATION COMPUTER RESOURCES (NGCR)

PROGRAM DESCRIPTION

NGCR PROGRAM PURPOSE

- PROVIDE COMPUTER RESOURCE STANDARDS CAPABLE OF MEETING NAVY MISSION CRITICAL REQUIREMENTS IN THE MID 1990'S AND BEYOND
  - HARDWARE AND SOFTWARE
  - SUCCESSOR TO CURRENT NAVY STANDARDS
  - COST EFFECTIVE

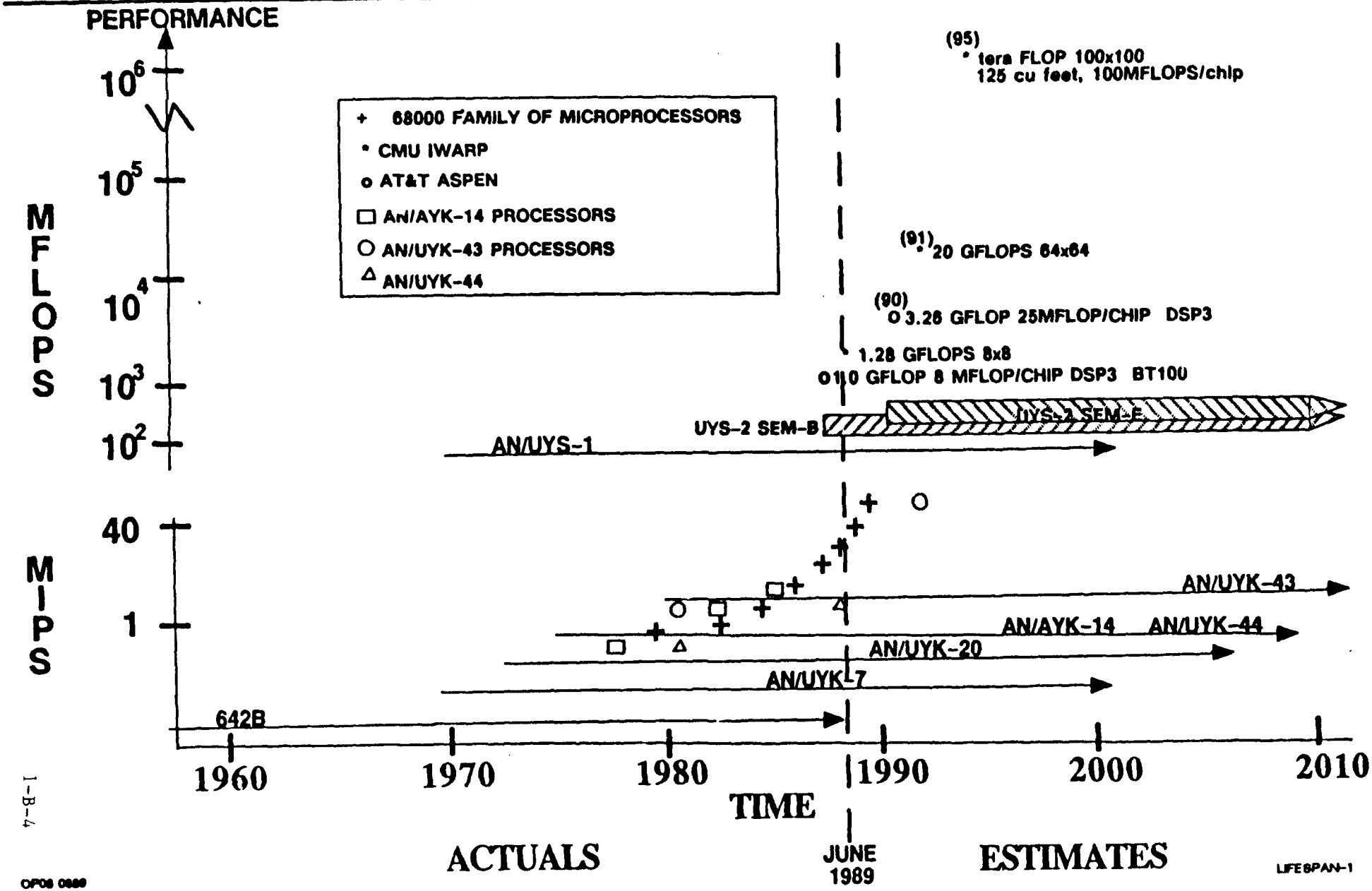
DOD SHARE OF THE MARKET – SEMICONDUCTOR PARTS



- NAVY CAN NO LONGER DICTATE MARKET
  - MUST PARTICIPATE WITH COMMERCIAL SECTOR
  - MUST LEVERAGE COMMERCIAL PRODUCT DEVELOPMENT

MARKET 84-1

STANDARD COMPUTER PERFORMANCE AND LIFE SPAN



OP06 0888

PROGRAM OBJECTIVES

- INCREASE WEAPONS SYSTEMS OPERATIONAL READINESS AND INTEROPERABILITY
- ADAPT TO RAPIDLY CHANGING THREAT
- PERMIT COMPETITION FOR PRODUCT DEVELOPMENT AND SYSTEM UPGRADES
- MEET WIDE RANGE OF APPLICATION REQUIREMENTS
  - AIR, SURFACE, SUB-SURFACE, LAND
  - ENVIRONMENT (COMMERCIAL - FULL MIL SPEC)
  - PROCESSING CAPABILITY
- HAVE AFFORDABLE COSTS
- PROVIDE THE FLEET THE MOST EFFECTIVE COMPUTER CAPABILITY THE NATION IS CURRENTLY (IN ALL DEPLOYMENT YEARS) PRODUCING
- INCREASE PROGRAM MANAGER'S FLEXIBILITY IN APPLYING / UTILIZING COMPUTER RESOURCE STANDARDS

NGCR CONCEPT

NGCR IS NOT

A COMPUTER SYSTEM

A SOFTWARE SYSTEM

NGCR IS

HARDWARE AND SOFTWARE
INTERFACE AND PROTOCOL STANDARDS

NGCR APPROACH
OPEN SYSTEMS ENGINEERING ARCHITECTURE

- PROVIDES FRAMEWORK FOR SYSTEMS DESIGN
  - DOES NOT DEFINE OR STANDARDIZE ON A COMPUTER DESIGN
  - STANDARDIZES HARDWARE / SOFTWARE INTERFACES
  - PROVIDES FRAMEWORK FOR INDUSTRY IR&D INVESTMENT
- FOLLOWS STANDARDIZATION TRENDS IN COMMERCIAL SECTOR
- IMPLEMENTED THROUGH JOINT NAVY / INDUSTRY WORKING GROUPS
  - WIDELY USED NON-PROPRIETARY COMMERCIAL STANDARDS BASE

STANDARDIZATION AREAS

MULTIPROCESSOR INTERCONNECTS

- BACKPLANE (1992)*
- HIGH SPEED DATA TRANSFER NETWORK (1994)
- HIGH PERFORMANCE BACKPLANE (1997)

MULTISYSTEM INTERCONNECTS

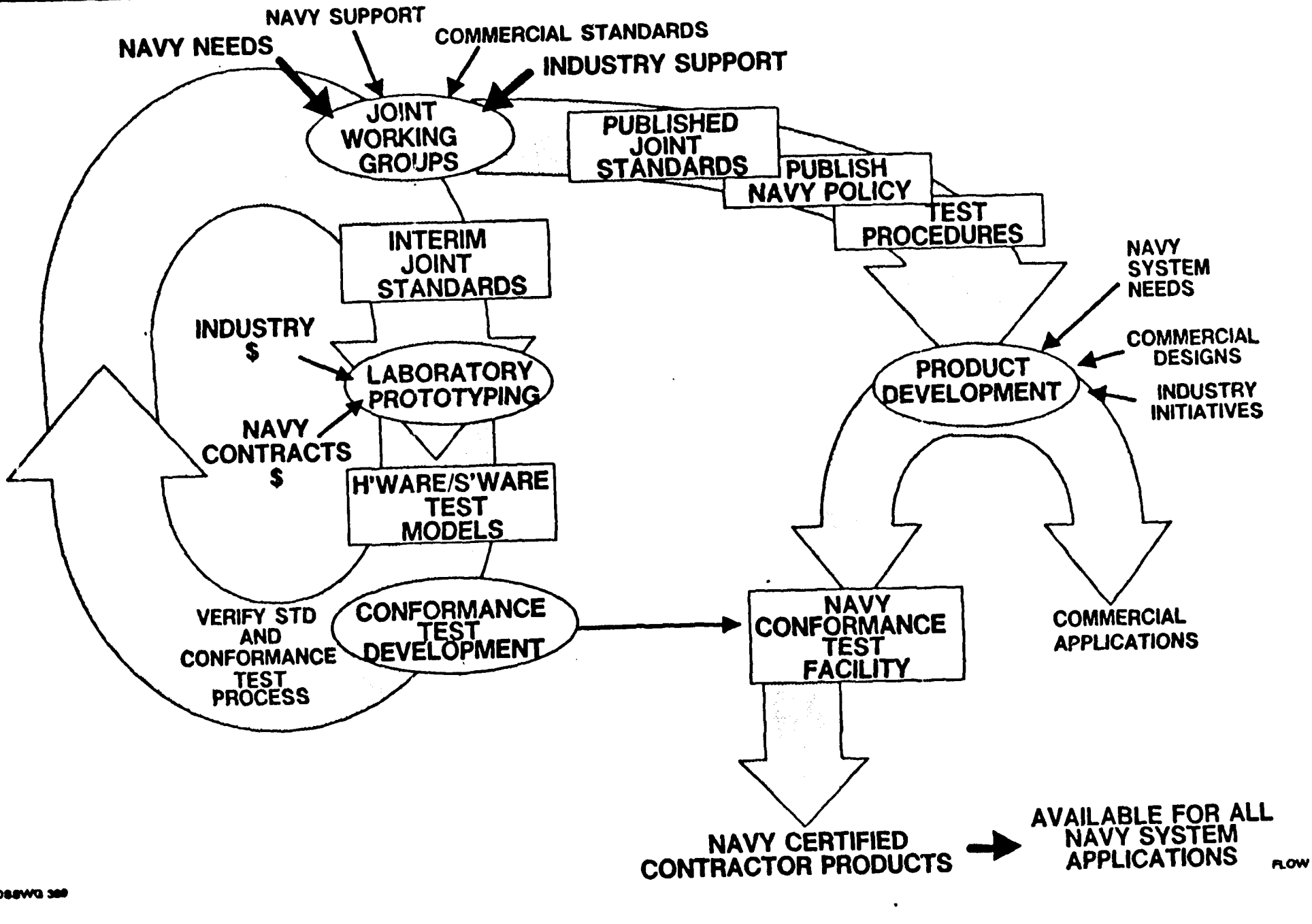
- SAFENET I / LOCAL AREA NETWORK (1990)*
- SAFENET II / LOCAL AREA NETWORK (1991)*
- HIGH PERFORMANCE LOCAL AREA NETWORK (1998)

SOFTWARE STANDARDIZATION AREAS

- OPERATING SYSTEM (1996*; 1998 MLS)
- DATA BASE MANAGEMENT SYSTEM (1998)
- PROGRAMMING SUPPORT ENVIRONMENT (1998)
- GRAPHICS LANGUAGE / INTERFACE (1998)

* CERTIFICATION OF PRODUCTS THROUGH NGCR

PROGRAM FLOW



OSSWG 389

NGCR STRUCTURE PERMITS :

KEY FACTORS

- COMPETITION
- INTEROPERABILITY
- COMMONALITY
- LOGISTICS
- SUPPORTABILITY
- TECHNOLOGY
- COMMERCIAL BASE
- MODULAR DESIGNS

- **RAPID CONTINUOUS INFLUX OF UP-TO-DATE COMPUTER RESOURCES**
  - COMPONENT TECHNOLOGIES
  - ARCHITECTURES
- **MUCH LARGER BASE OF COMPETITIVE SOURCES**
  - POSITIVE COST / SCHEDULE / PERFORMANCE CONSIDERATIONS
  - NON-PROPRIETARY OSA INCREASES COMPETITION FOR SYSTEMS DEVELOPMENTS AND UPGRADES THROUGHOUT LIFE CYCLE
- **LEVERAGING OF COMMERCIAL BASE**
  - INDUSTRY INVESTMENT
  - COMPETITION
  - INNOVATION
- **INTEROPERABILITY OF MULTI-VENDOR NGCR COMPLIANT PRODUCTS**
- **COMMONALITY OF NGCR PRODUCTS TO REDUCE LOGISTICS COSTS**
- **MODULAR / ADAPTABLE SYSTEMS DESIGNS**
  - RAPID / COMPETITIVE "PLUG-IN" UPGRADES
  - BUILDING BLOCK DESIGNS FOR FULL SPECTRUM OF PROCESSING CAPABILITIES
  - TRUE EVOLUTIONARY DEVELOPMENTS
- **PROGRAM MANAGER FLEXIBILITY**
  - SYSTEM DESIGN
  - ACQUISITION MANAGEMENT
- **TOP DOWN WEAPONS SYSTEM DESIGN**

SAFENET WORKING GROUP

SAFENET I / II PHYSICAL CHARACTERISTICS

	SAFENET I	SAFENET II
BASED ON	IEEE 802.5 (802.5C)	ANSI X3T9.5 FDDI
SUPPORTED BY	TMS-380 CHIP SET	AMD FDDI CHIP SET
TOPOLOGY	DUAL TOKEN RING	DUAL TOKEN RING
BUS RATE	16 MBPS	100 MBPS
MAX DISTANCE BETWEEN USERS	300 M	2000 M
MAX RING CIRCUMFERENCE	10 KM	100 KM
MAXIMUM NUMBER OF USERS	128	500

SAFENET / LAN STANDARDS WORKING GROUP

* GOVERNMENT

AFSTC	JDL	NAVAIR	NCSC	NSWC
AIRMICS	NAC	NAVDAC	NESESA	NUSC
CPM	NADC	NAVSEA	NOSC	PMTC
FCDSSA/SD	NASA	NBS	NRL	SPAWAR
				USCG

* PRIVATE INDUSTRY/ACADEMIA

ADSI	ESL	MAGNAVOX	SEMCOR
AMD	FAIRCHILD	MARTIN MARIETTA	SIECOR
AMP	FERRANTI	MITRE	SILICON GRAPHICS
ARINC	FMC	NORTH ATLANTIC	SPAR
ARNOLD ASSOC.	G&H TECHNOLOGY	NORTHROP	SPERRY MARINE
ASD	GENERAL DYNAMICS	OCEAN TECHNOLOGY	SYNETICS
AT&T	GE/RCA	ORYX	TECHREP/SYSCON
BGS	GOULD	PCO	TEXAS INSTRUMENTS
BIT	GRUMMAN	PLESSEY ELECT SYS	UNISYS
BOOZ, ALLEN	GTE	PROTEON	UNIV OF VIRGINIA
CIRCUIT TECHNOLOGY	HONEYWELL	PROTOCOL ENGINES	VANCE
CMU	HUGHES	RAYCHEM	VAN DYKE ASSOC.
CONTROL DATA	IBM	RAYTHEON	VITRO
CSC	ITT	ROCKWELL	WESTINGHOUSE
CTI	JHU/APL	ROLM	XEROX
ELDYNE	LITTON DSD	SANDERS ASSOCIATES	

Survivable Adaptable Fiber Optic Embeddable Network (SAFENET)

SAFENET LAN WORKING GROUP: **CO CHAIRMAN:** LCDR RON OWENS
202-692-3966

ED HOWARD, NOSC
619-553-3403

SAFENET SUBGROUP ACTIVITIES:

PHYSICAL MEDIA: LT JEFF PAIGE, NOSC
619-553-3413

COMM SERVICES: DAVE MARLOW, NSWC
703-663-1675

NETWORK MGT: DAN GREEN, NSWC
703-663-1073

MEETINGS

01/23/90-01/26/89: San Diego, CA

03/20/90-03/22/90: Washington, DC

05/23/90-05/25/90: Newport, RI

07/24/90-07/26/90: Washington, DC

09/11/90-09/13/90: San Diego, CA

11/06/90-11/08/90: Washington, DC

MILESTONES

JAN 90: SAFENET I Standard Completed

MAR 90: SAFENET II Draft (1) Release

SEP 90: SAFENET II Technical Freeze

JAN 91: SAFENET II Standard Completed

SEP 91: MIL-STD SAFENET I Completed

NOV 91: SAFENET III (HP LAN) Start

SEP 92: MIL-STD SAFENET II Completed

SN SUMMARY

OPERATING SYSTEM WORKING GROUP

INITIAL TECHNOLOGIES LIST

STANDARDS:

CAIS-A
MAP
ORKID
OSF/I
OSI
POSIX
RTEID

SYSTEMS:

ACCENT
ALPHA
ALS/N
AMOEBA
ARGUS
ARTS
ARTX
ASOI
ASOS
ATES
BFS
BIIN OS
BSO
CCIU-OS
C-EXECUTIVE
CHAOS/GEM
CHOICES
CLOUDS
CMP/OS
CO-IDRIS
CRONUS
CRYSTAL
CXOS
DARK
DINOS
D-NIXOS
DOSK
DRAGON/MELODY
E10.S
EDEN
ELXSI
FLEXOS
43RSS
GALAXIE
GUARDIAN (H.)
GUARDIAN (T.)
HARMONY
HARTOS
HERBER-II
HOPS
HP-UX
HYPERFLOW
HXDP
IDRIS
IRMX
ISIS
KIT
KOCOS
KSOS
LOCUS
LYNXOS
MACH
MARUTI
MIKE
MOSI
MEDUSA
MFM
MICROS
MIMAS
MTOS
MUNET
OS-9
PAVE PILLAR
PDOS
PHIDIAS
PHOENIX PSOS
QNX
REBUS
REGULUS
RIG
RMS68K
RSS/M
RTEID
RTU
SCOMP
SDX
SDEX/20
SDOS
SHOSHIN
SIRIUS-DELTA
SODS/OS
SPOX
SPRING KERNEL
SPRITE
STARLITE
STAROS
TCNA
TELESOFT
T-MACH
TRON
UNIFLEX
V-SYSTEM
VERSADOS
VEXTRA
VRTX
VXCEL
VXWORKS
WISOS
XOS
XTS-200
ZCZOS
ZMOB-OS

TECHLST-11/89

CANDIDATE OS INTERFACE STANDARDS/SYSTEMS

NAME	SPONSOR/COMPANY	DESCRIPTOR
1. ALPHA	DARPA/CONCURRENT CORP.	DISTRIBUTED, RT KERNEL
2. ARTX	READY SYSTEMS	DISTRIBUTED,RT EXEC.
3. CRONUS	RADC/BBN	DISTRIBUTED OS
4. IRMX	INTEL CORP.	OS KERNEL
5. MACH	DARPA/CMU/OSF	DISTRIBUTED OS
6. MTOS	INDUSTRIAL PROGRAMING INC.	RT KERNEL
7. ORKID	VITA	RT KERNEL INTERFACE
8. POSIX	IEEE	OS INTERFACE STANDARD
9. SDOS	DARPA/ODYSSEY RESEARCH	SECURE DISTRBUTED OS
10. TRUSTED MACH	DARPA/TRUSTED INFORMATION SYTEMS INC.	SECURE DISTRIBUTED OS

OS CANDIATES

OSSWG FUNCTIONAL REQUIREMENTS

• SERVICE CLASS

- **GENERAL REQUIREMENTS**
  - SCOPE(NAVY APPLCTIONS)
  - DESIGN OBJETIVES
    - COMPATIBILITY
    - MAINTAINABILITY
    - INTEROPERABILITY
    - TRANSPORTABILITY
    - REUSABILITY
  - BASIC SERVICES
  - ARCHITECTURE INDEPENDENCE
  - MODULARITY
  - EXTENSIBILITY
  - UNIFORMITY
  - COMPLETENESS
- **ARCHITECTURE-DEPENDENT**
  - FUNCTIONALITY SUBSET
    - DOWNLOAD
    - INITIALIZATION
    - START
    - STOP
    - SHARE RESOURCES
    - PROCESS TO PROCESS COM
    - OPS STATUS INFO
- **CAPABILITY & SECURITY**
  - SECURITY POLICY
    - DISCRESIONARY ACCESS CONTROL(DAC)
    - OBJECT REUSE
    - LABELS
    - LABEL INTEGRITY
    - LABEL INFO EXPORTATION
    - EXPORTATION MULTI-LEVEL; DEVICES
    - SINGLE LEVEL
    - LABEL HUMAN READ-ABLE OUTPUT
    - SENSITIVITY LABELS
    - DEVICE LABELS
    - MANDATORY ACCESS CONTROL(MAC)
    - ACCOUNTABILITY
    - TRUSTED PATH
    - AUDIT
    - ASSURANCE
    - DOCUMENTATION
- **DATA INTERCHANGE**
  - SERVICES FOR
    - DATA FORMAT CONVERSION
    - BETWEEN COM-PILERS &/OR CPUs
- **EVENT & ERROR MGMNT**
  - EVENT/ERROR COLLECTION
  - EVENT/ERROR DISTRIBUTION
  - EVENT/ERROR FILTERING & PROCESSING ALTS
  - LOGGING
  - ENABLE/DISABLE INTERRUPTS
- **FILES**
  - CONTIGUOUS ALLOCATION
  - PROTECT AREA
  - INTER-PROGRAM COMMS
  - SCHEDULING (RESPONSE)
  - SUSPEND/CONTINUE DELAY
  - BLOCK REQUESTS
  - ROUND ROBIN RECORD OVERLAY
  - SERVICES: CREATE,OPEN,POINT
  - QUERY,READ,MODIFY,CLOSE
  - DIRECTORY SERVICES, SHADOWS
- **GENERALIZED I/O**
  - DEVICE DRIVER SUPPORT
  - DEVICE SERVICES
    - OPEN
    - CLOSE
    - TRANSMIT DATA
    - RECEIVE DATA
    - EVENT NOTIFY
    - CONTROL
    - MOUNT/DISMOUNT
    - SUSPEND/CONTINUE
    - I/O DIRECTORY SVCS
- **NETWORK & COMMS**
  - INTERFACES TO CONTROL OF
    - FUTUREBUS+
    - SAFENET
    - MIL-STD-1553B
    - MIL-STD-1397B
  - SUPPORT INTERFACES TO & CONTROL OF OTHER NETWORK & COMMUNICATION ENTITIES

OSREQMT81

OSSWG FUNCTIONAL REQUIREMENTS

• SERVICE CLASS (CONT.)

• PROCESS MANAGEMENT

PROCESS SERVICES:
- CREATE
- START
- TERMINATE
- SUSPEND
- RESUME
- DELAY
- SAVE/RESTART

INTERPROCESS COMMUNICATION(IPC)
EXAMINE/MODIFY PROCESS ATTRIBUTES
EXAMINE STATUS
PROCESS ID
MULTIPROGRAMMING SUPPORT

• PROJECT SUPPORT ENVIRONMENT INTERACTION

DEBUG SERVICES:
- EXAMINE/ALTER REGISTRS
- SET/CLEAR BRKPNTS & WATCHPONTS
- SINGLE-STEP EXECUTION
- CONTINUE EXECUTION
- EXAMINE/ALTER MEMORY
- QUERY PROCESS ENVIRONMENT
- QUERY CALL STACK
- EXECUTION PROFILE

RELIABILITY, ADAPTABILITY & MAINTAINABLTY

FAULT INFORMATION COLLECTION, REQUEST
DIAGNOSTIC TEST REQST
SELF TEST RESULTS
DIAGNOSTIC TST RESULTS
QUERY OPERATIONAL STATUS
FAULT DETECTION THRESHOLDS
FAULT ISOLATION
FAULT RESPONSE
RECONFIGURATION
ENABLE/DISABLE COMPONENT
PERFORMANCE MONITORING
RESOURCE UTILIZATION LIMITS
CHECKPOINT DATA STRUCTURES

RESOURCE MNGMNT

VIRTUAL MEMORY
VIRTUAL SPACE LCKG
DYNAMIC MEMORY ALLOC/DEALLOC
QUERY/SET MEMORY PROTECT ATTRIBTS
SHARED MEMORY
DEVICE ALLOCATION/DEALLOCATION
REMOVABLE MEDIA MOUNTING/DISMNT
DESIGNATION/RELEAS RESOURCE STS & CONFIG RESPNSBTY
PROCESSOR ALLOCATION/DEALLOCA
SYSTEM RESOURCE RQMNTS SPEC
QUERY SYSTEM RESOURCE CAPCTY

• SYNCHRONIZTION/SCHED

PROCESS SYNCHRNZATION
SCHEDULING DELAY
PERIODIC SCHEDULING
MULTIPLE SCHED POLICYS
SELECTION SCHD POLICY
MOD OF SCHED PARAMS
PRECISE SCHEDULING
MUTUAL EXCLUSION
CUMULATIVE PROC EXEC
ATTACH PROCESS TO EVNT
TRANSACTN SCHED INFO

• SYSTEM INITIALIZATION & REINITIALIZATION

PROGRAM LOAD (LOCAL & REMOTE)
SYSTEM INITIALIZATION

• TIME SERVICES

READ SELECTED CLOCK
SET SELECTED CLOCK
SYNCHRONZTN OF CLOCKS
SELECT PRIMARY REF CLCK
LOCATE PRIMARY REF CLCK
TIMER SERVICES:
- SET ALARM
- CLEAR ALARM
- NOTIFICATION ALRM TM

PRECISION CLOCK (TIME GRANULARITY)

• ADA LANGUAGE SUP

SUP TASK CREATION
SUP TASK ABORTION
SUP TASK SUSPEND
SUP TASK RESUME
SUP TASK TERMINATE
SUP TASK RESTART
SUP ACCESS TSK CHR
SUP TSK EXEC MONIT
"ACCSS PRECS RT-CLK
"ACCSS TIME OF DAY CLOCK ETC. OSREQMT32

OPERATING SYSTEMS STANDARDS WORKING GROUP PARTICIPANTS

* PRIVATE INDUSTRY/ACADEMIA

ADI
ARNOLD ASSOC.
ARINC RESEARCH CORP.
AT&T
BBN SYS & TECH CORP.
BENDIX FLD ENG CORP.
BOOZ, ALLEN
CEA INCORPORATED
CMU
COMPUNETICS, INC.
COMPUTER-BASED SYS
CONCURRENT COMPTR
CONTROL DATA CORP
CRAY RESEARCH, INC.
CSC
CTA
CYBERCHRON CORP.
DATA GENERAL CORP.
DECISION SYS ENG
DRC
DY-4 SYSTEMS INC.
EDS
EL-TECH SUPPORT
GENERAL DYNAMICS
GENERAL ELECTRIC
GEORGIA TECH
HARRIS GCSD
HEWLETT PACKARD
HONEYWELL
HUGHES
IBM
INTEL
INDUSTRIAL PROG
IDA
INTEGRATED MICRO
JHU/APL
LOCKHEED
MCDONNELL DOUGLAS
MITRE
NUMERIX CORP
ODYSSEY RESCH ASSOC.
PRC
RAYTHEON
READY SYSTEMS
ROCKWELL
SEMCOR
SIGMA PLUS, INC.
SOFTECH, INC.
SYSCON
SYSTEMS EXPLOR INC
TELEDYNE SYSTEMS
TELEDYNE BROWN ENG
TEXAS INSTRUMENTS
TRW
UNISYS
UNIV OF VIRGINIA
UNIV OF MARYLAND

* GOVERNMENT

DONIRM
FAA
FCDSSA/SD
FCDSSA/DN
ONR
NAC
NADC
NASA
NATC
NAVAIR
NAVSEA
NIST
NOSC
NRL
NSWC
NUSC
NWC
NWSC-CRANE
RADC-USAF
SPAWAR
USCG

OPERATING SYSTEMS STANDARDS WORKING GROUP (OSSWG)

OSSWG: CO-CHAIRMEN: CDR RICK BARBOUR
202-692-3966

TRICIA OBERNDORF, NADC
215-441-2737

OSSWG SUBGROUP ACTIVITIES:

REQUIREMENTS: RICH BERGMAN, NOSC
619-553-4098

AVAILABLE TECHNOLOGY: PHIL HWANG, NSWC WHITE OAK
202-394-1351

APPROACH: TOM CONRAD, NUSC
401-841-3353

MEETINGS

08/01/89-08/03/89: Washington, DC
09/12/89-09/14/89: Washington, DC
10/17/89-10/19/89: Newport, RI
12/05/89-12/07/89: San Diego, CA
01/22/90-01/26/90: Mobile, AL
03/06/90-03/08/90: Washington, DC
04/17/90-04/19/90: Pittsburgh, PA

MILESTONES

Oct 89: Technology Report
May 90: Operational Concept Document
Jan 91: Begin Interim Standard
Dec 93: Interim Standard
Dec 95: Final Standard

LABORATORY TEST MODELS CONTRACTS

BACKPLANE, SAFENET I & II

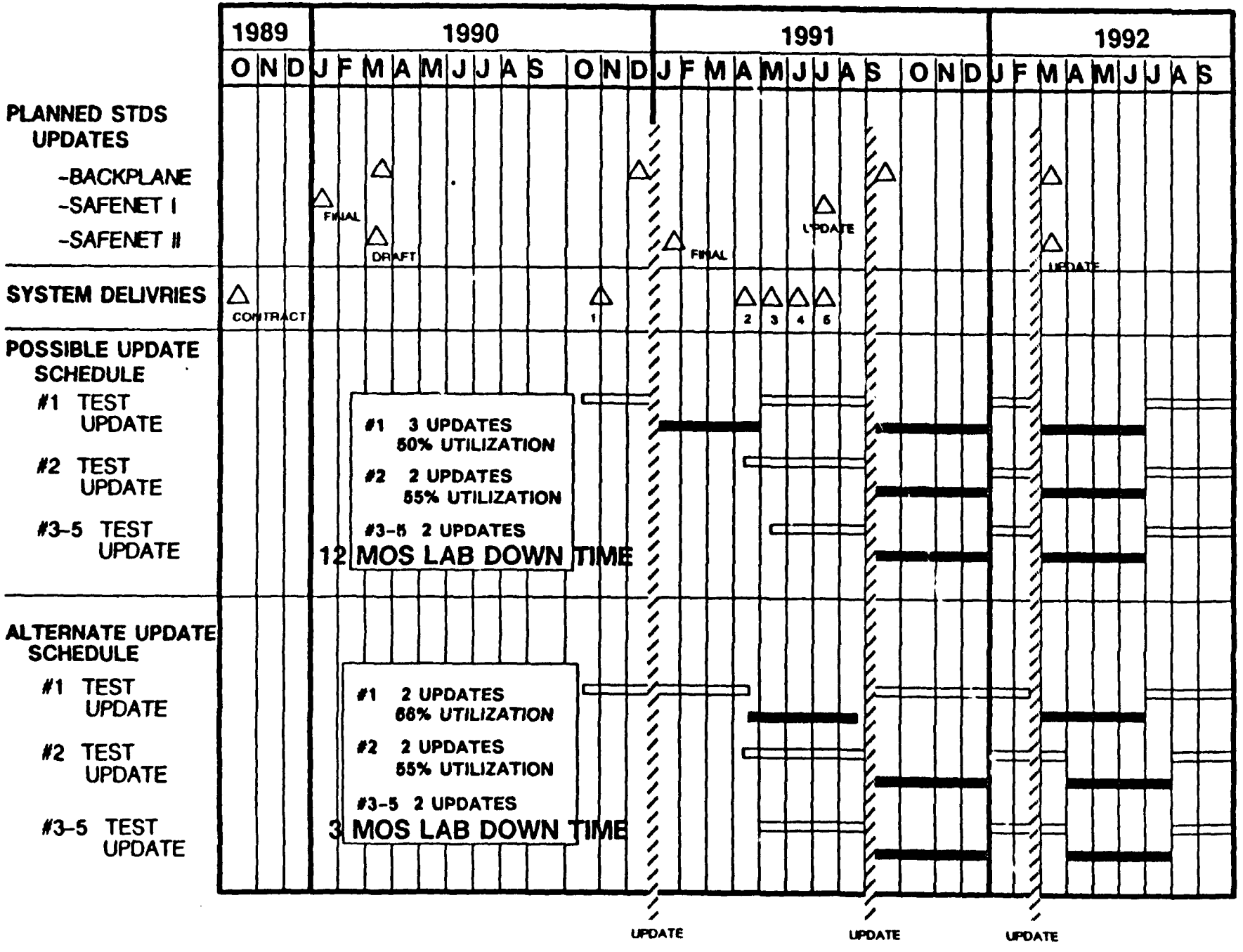
PROTO COVER

CONTRACTOR DELIVERABLES

- 5 SYSTEMS EACH
- FIRST DELIVERIES – NOVEMBER 1990

COMPANY	CPU #1	CPU #2	SAFENET I & II	FUTUREBUS+	8 MBYTES MEMORY CABINET & POWER SUPPLY	I/O MIL-STD-1553B 1397 NTDS RS-232
RAYTHEON	VAX	MIPS R3000	√	√	√	√
LITTON DATA SYSTEMS	INTEL 80486	MOTO. 88000	√	√	√	√
CABLE & COMPUTER TECHNOLOGY	MOTO. 68030	AMD AM29000	√	√	√	√

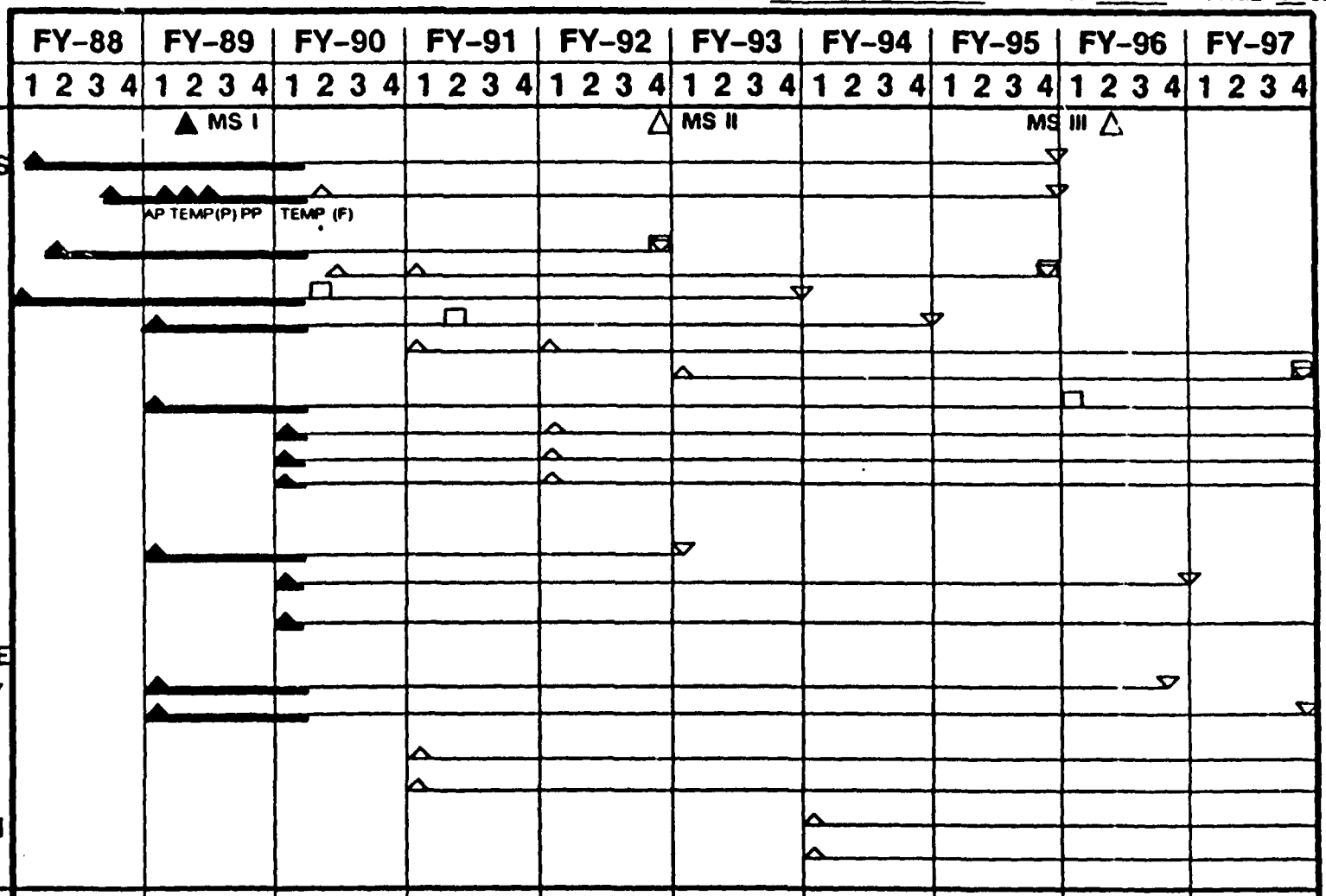
UPDATES FOR STANDARDS COMPATIBILITY



PROGRAM SCHEDULE

TIER 1 REV R 8 DATE 1 JAN 90

APPROVED _____ DATE _____ PAGE 1 OF 1



WBS ELEMENT ($M)

- 1.0 ENGR. STUDIES
- 2.0 PGM. DOCMNT
- 3.0 STANDARDS
  - 3.1 BACK PLANE
  - 3.2 SWITCH
  - 3.3 LAN SAFENET I
  - 3.4 LAN SAFENET II
  - 3.5 HP LAN
  - 3.6 HP BACKPLANE
  - 3.7 OPERATING SYS
  - 3.8 DBMS
  - 3.9 PSE
  - 3.10 GRAPHICS
- 4.0 PROTOTYPES
  - 4.1 BACK PLANE
  - 4.2 OPERATING SYS
  - 4.3 APPLICATIONS
- 5.0 CONFORMANCE
  - 5.1 METHODOLOGY
  - 5.2 FACILITY
- 6.0 ILS
- 7.0 STDS MAINT
- 8.0 CERTIFICATION
- 9.0 TEST BED

		FY-88	FY-89	FY-90	FY-91	FY-92	FY-93	FY-94	FY-95	FY-96	FY-97
1-5 RDT&E	FYDP	1.1	3.7	12.2	14.0	29.1	28.7	32.6	33.1	33.7	34.3
	REQ	1.1	4.7	12.2	14.0	31.6	31.1	37.0	29.4	17.7	13.4
6-8 O&M,N	FYDP	–	–	–	1.1	1.3	1.4	2.0	2.1	2.1	2.2
	REQ	–	–	–	1.1	1.3	1.8	2.6	7.4	8.1	5.6
9 OPN	FYDP	–	–	–	–	–	–	2.4	2.4	2.4	2.5
	REQ	–	–	–	–	–	–	2.4	5.1	6.2	6.2
TOTAL FYDP		1.1	3.7	12.2	15.1	30.4	30.1	37.0	37.6	38.2	39.0
TOTAL DELTA		0	0	0	0	+2.5	+2.8	+5.0	+4.3	-6.2	-13.8

MAJOR ACTIVITIES

FY-89 ACCOMPLISHMENTS

- INITIATED CERTIFICATION TEST PLANS AND PROCEDURES
- INITIATED MIL SPEC FOR RUGGEDIZED EQUIPMENT ENVIRONMENT
- INITIATED INTEGRATED LOGISTICS SUPPORT PLAN
- INITIATED INDUSTRY/NAVY WORKING GROUPS
  - SAFENET II LOCAL AREA NETWORK STANDARD
  - OPERATING SYSTEM STANDARDS
- INITIATED LABORATORY TEST MODEL CONTRACT EFFORTS
  - BACKPLANE, SAFENET I, SAFENET II
  - OPERATING SYSTEM
- CONTINUED INDUSTRY/NAVY WORKING GROUPS
  - SAFENET I
  - BACKPLANE
- PUBLISHED DRAFT SAFENET I INTERFACE STANDARD
- COMPLETED PROGRAM DOCUMENTATION
  - PROGRAM MASTER PLAN
  - ACQUISITION PLAN
  - TEST AND EVALUATION MASTER PLAN
- MILESTONE I APPROVAL TO PROCEED

MAJOR ACTIVITIES

FY-90 PLANS

- INITIATE SAFENET II INTERIM CERTIFICATION TESTING (NCCS AFLOAT)
- INITIATE ANALYSIS FOR
  - HIGH SPEED DATA TRANSFER NETWORK STANDARDS
  - DATA BASE MANAGEMENT SYSTEM STANDARDS
  - PROGRAMMING SUPPORT ENVIRONMENT STANDARDS
  - GRAPHICS INTERFACE STANDARDS
- CONTINUE INDUSTRY/NAVY WORKING GROUPS
  - BACKPLANE
  - OPERATING SYSTEMS
  - SAFENET I & II
- CONTINUE CERTIFICATION TEST PLANS AND PROCEDURES
- CONTINUE MIL SPEC FOR RUGGEDIZED EQUIPMENT ENVIRONMENT
- CONTINUE ILSP
- AWARD OPERATING SYSTEM CONTRACTS (ENGINEERING ANALYSIS)
- AWARD MULTIPLE CONTRACTS
  - LABORATORY TEST MODELS (BACKPLANE, SAFENET I&II)
- PUBLISH SAFENET I INTERFACE STANDARD AND HANDBOOK
- PUBLISH DRAFT INTERFACE STANDARDS
  - BACKPLANE
  - SAFENET II

SUMMARY

KEY FACTORS

- COMPETITION
- INTEROPERABILITY
- COMMONALITY
- LOGISTICS
- SUPPORTABILITY
- TECHNOLOGY
- COMMERCIAL BASE
- MODULAR DESIGNS

- REVOLUTION IN NAVY COMPUTER RESOURCES ACQUISITION STRATEGY OF PAST 30 YEARS TO REFLECT DOD MARKET SHARE AND LONG DEVELOPMENT CYCLE
  - NON-PROPRIETARY OPEN SYSTEMS ARCHITECTURE
  - CFE
- EMPHASIS ON COMPETITION, INTEROPERABILITY AND COMMONALITY
- RAPID, CONTINUOUS TRANSITION OF U.S. INDUSTRY STATE OF THE PRACTICE TO NAVY SYSTEMS
  - COMMERCIAL DESIGNS AND WIDELY ACCEPTED NON-PROPRIETARY COMMERCIAL STANDARDS
  - INDUSTRY DESIGN CREATIVITY AND TECHNOLOGIES

BACKPLANE WORKING GROUP

FUTUREBUS+ ATTRIBUTES

CORESIDENT CACHE AND MESSAGE PASSING SCHEMES PERMIT TIGHTLY COUPLED "SUB-SYSTEMS" TO BE LOOSELY COUPLED TO OTHER SUB-SYSTEMS IN THE SAME ENCLOSURE

SPLIT TRANSACTION CAPABILITY PROVIDED FOR MULTI-BUS SYSTEMS

PROTOCOLS SUPPORT LIVE INSERTION

PARITY PROTECTION PROVIDED ON ADDRESS/DATA, COMMAND, STATUS, ARBITRATION, AND TAG LINES

PACKET MODE TRANSACTIONS (WITH HANDSHAKE AT END OF PACKET) PERMITS FASTEST POSSIBLE TRANFER CAPABILITY

PERFORMANCE CAPABILITIES :

YEAR	SPEED	@ 32 BITS	@ 64 BITS	@ 128 BITS	@ 256 BITS
1990	25 MT/SEC	100 MB/SEC	200 MB/SEC	400 MB/SEC	800 MB/SEC
1992	50 MT/SEC	200 MB/SEC	400 MB/SEC	800 MB/SEC	1600 MB/SEC
1994	100 MT/SEC	400 MB/SEC	800 MB/SEC	1600 MB/SEC	3200 MB/SEC

MT/SEC = 10^6 TRANSACTIONS/SEC MB/SEC = 10^6 BYTES/SEC

10/89

FUTUREBUS+ ATTRIBUTES

EXPECTED TO BE THE ONLY WIDELY USED (SUPPORTED BY USN, VITA, MMG, ...) NON-PROPRIETARY, HIGH PERFORMANCE BACKPLANE BUS

A 64 BIT (ADDRESS/DATA) ARCHITECTURE WITH A COMPATIBLE 32 BIT SUBSET AND DATA EXTENTION CAPABILITY TO 128 AND 256 BITS

DESIGNED TO BE PROCESSOR, TECHNOLOGY, AND ARCHITECTURE INDEPENDENT

THE SYNCHRONOUS, COMPELLED, THREE WIRE HANDSHAKE PERMITS TRANSACTIONS TO OCCUR AS FAST AS THE PARTICIPANTS ARE CAPABLE

HANDSHAKING ON BOTH THE LEADING AND TRAILING EDGES OF THE SYNCHRONIZING LINES DOUBLES THE PERFORMANCE OF PREVIOUS BUSES

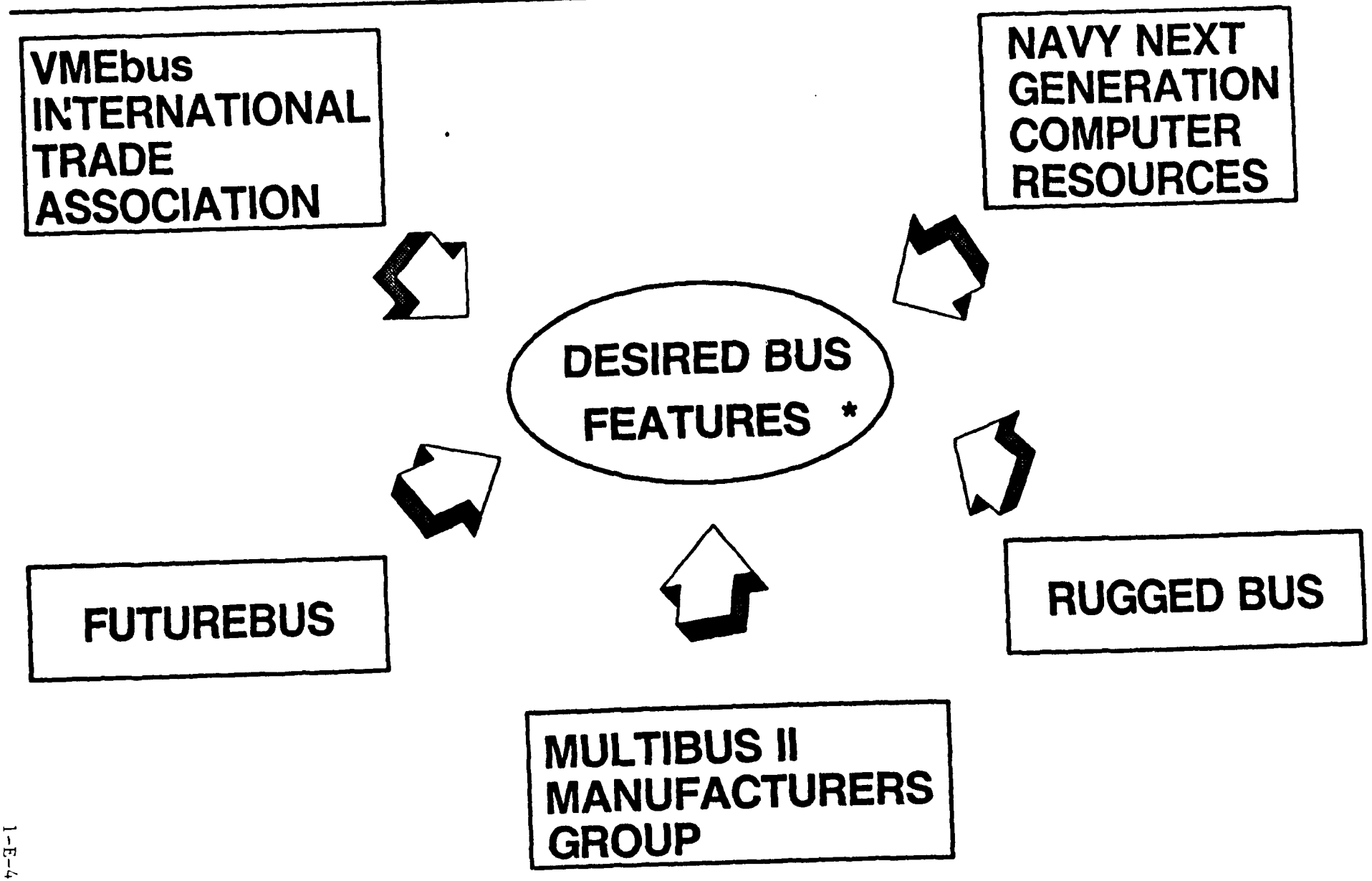
BACKPLANE TRANSCEIVER LOGIC (IEEE P1394) PROVIDES INCIDENT WAVE SWITCHING, SOLUTION TO THE BUS DRIVING PROBLEM

CACHE COHERENCY PROTOCOLS (SNOOPING, SNARFING, INTERVENTION WITH REFLECTION) PROVIDE HIGH PERFORMANCE SHARED MEMORY MULTI-PROCESSING CAPABILITY

LAYERED MESSAGE PASSING PROTOCOL IS CAPABLE OF SUPPORTING MULTIPLE MESSAGE FORMATS. MESSAGE MAILBOX IN CSR SPACE DISTINGUISHES MESSAGES FROM OTHER TYPES OF TRANSACTIONS. REQUEST AND RESPONSE MAILBOXES SUPPORT SOLICITED AND UNSOLICITED MESSAGES.

10/89

BACKPLANE BUS REQUIREMENTS ANALYSIS



* THIS WILL BE THE FUTUREBUS+ STANDARD

BACKPLANE STANDARDS WORKING GROUP

COMPANIES SUPPORTING WORKING GROUP :

AITECH
AMERICAN SYSTEMS CORP
AMERITECH SERVICES
AMP, INC
AMPERIF CORPORATION
ANALYSIS & TECHNOLOGY
APTEC
ARINC RESEARCH CORP
AT&T
BOOZ-ALLEN
CENTRAL DATA CORP
CONTROL DATA CORP
DATA GENERAL CORP
DELCO
DGA, INTERNATIONAL
DIGITAL EQUIPMENT CORP
DY-4 SYSTEMS
EG & G
ELECTRONIQUE SERGE DASSAULT
FORCE COMPUTERS
GENERAL DYNAMICS
GRUMMAN
HEWLETT PACKARD
HONEYWELL
HUGHES
IBM
INTEL
INTERSTATE ELECTRONICS
JOHN HOPKINS UNIVERSITY
LOGICON
LORAL
MARKEN
MARTIN-MARIETTA
MCC
MICROBAR
MOTOROLA
NATIONAL SEMICONDUCTOR
NORDEN SYSTEMS
ORI
PLANNING RESEARCH CORP
PLESSEY ELECTRONICS
RADSTONE TECHNOLOGY
RAYTHEON
ROLM
RUGGED DIGITAL
SEI/CMU
SINGER
SYNETICS
TANDEM COMPUTERS
TASC
TELEDYNE SYSTEMS
TEREDYNE CONNECTION SYS
TEXAS INSTRUMENTS
TIBURON
TITAN-SESCO
TRW
UNISYS
VITA
VITRO
WESTINGHOUSE

NAVY / MILITARY SUPPORT :

SPAWAR
NAVSEA
NAVAIR
NAVAL AVIONICS CENTER
NAVAL AIR DEVELOPMENT CENTER
NAVAL AIR TEST CENTER
NAVAL OCEAN SYSTEMS CENTER
NAVAL RESEARCH LABORATORY
NAVAL SURFACE WARFARE CENTER
NAVAL UNDERSEA SYSTEMS CENTER
NAVAL WEAPONS CENTER
NAVAL WEAPONS SUPPORT CENTER
U. S. COAST GUARD

RECEIVED INQUIRIES FROM THREE ARMY SOURCES

Navy Backplane, IEEE P896 (Futurebus+)

DOCUMENTS BEING DEVELOPED

MIL-HANDBOOK MCCR 00038

IEEE 896.1 Futurebus+ Logical Layer

IEEE 896.2 Futurebus+ Physical Layer

IEEE 896.3 Futurebus+ Systems Configuration

IEEE 1212 Control and Status Registers

IEEE 1101.n Core Mechanical Specification

IEEE 1156 Environmental Specification

MILESTONES

JAN 90: Futurebus+ Frozen (IEC, JTC, Public Review)

MAR 90: Navy Backplane Draft, SYSCOMS Review

MAY 90: Interim Standard Release, Initial Navy Use

MAY 90 - DEC 91 Prototype Updates to Standard

JUL 90: Futurebus+ Sponsor Ballot Begins

DEC 90: Futurebus+ Approved by IEEE

MAR 92: Navy Mil-Handbook, Final

MEETINGS

01/15/90-01/19/90: Lake Tahoe, CA

03/12/90-03/16/90: San Jose, CA

05/14/90-05/18/90: United Kingdom

07/19/90-07/20/90: Sun Valley, ID

09/17/90-09/21/90: Boston, MA

WORKING GROUP CHAIRMEN :

LCDR HARRISON BEASLEY 703-602-3966

RALPH LACHENMAIER, NADC 215-441-1634

IEEE 896 CHAIRMAN : DR PAUL BORRILL
SUN MICROSYSTEMS, 415-688-9187

896.1 VICE-CHAIR : DR. PAUL BORRILL

896.2 VICE-CHAIR : LCDR BEASLEY

896.3 VICE-CHAIR : ED JACQUES, JHU / APL
301-953-6327

1/90

Part 2

Record of Progress

The Operating Systems Standards Working Group (OSSWG) conducted fourteen meetings from January 1989 through August 1990. A brief schedule of the location and dates of those meetings is listed below. The official minutes of each of these meetings is included in this section. The presentation slides for each meeting will not be included however, but may be obtained by contacting LCDR Robert Voigt, SPAWAR 324A, Washington DC. (703) 602-3966.

In addition, video tapes have been made of each OSSWG meeting except the January 1989 meeting. These will be made available by OSSWG and a copy can be obtained by again contacting LCDR Robert Voigt at the number stated above.

Navy planning meeting:	NADC, Warminster, PA	18-19 Jan 1989
1st public meeting:	NAVSWC, White Oak, MD	16-17 Mar 1989
2nd meeting:	NAVSWC, White Oak, MD	16-18 May 1989
3rd meeting:	Vitro Corp. and Booz-Allen & Hamilton, Inc., Crystal City, MD	20-22 Jun 1989
4th meeting:	NAVSWC, White Oak, MD	1-3 Aug 1989
5th meeting:	NAVSWC, White Oak, MD	12-14 Sep 1989
6th meeting:	NUSC, Newport, RI	17-19 Oct 1989
7th meeting:	Vacation Inn, San Diego, CA	12-14 Dec 1989
8th meeting:	Stouffer Riverview Plaza Hotel Mobile, AL	22-26 Jan 1990
9th meeting:	NAVSWC, White Oak, MD	6-8 Mar 1990
10th meeting:	SEI, Pittsburgh, PA	17-19 Apr 1990
11th meeting:	NAVSWC, White Oak, MD	5-7 Jun 1990
12th meeting:	In conjunction with the IEEE POSIX meeting, Danvers, MA	16-20 Jul 1990
13th meeting:	NAVSWC, White Oak, MD	28-30 Aug 1990

NEXT GENERATION COMPUTER RESOURCES (NGCR) MEETING

January, 1989

ATTENDEES:

(Co-Chair OSSWG)	Rick Barbour	SPAWAR
	Rich Bergman	NOSC
	Dale Brouhard	NOSC
	Stephen Cecil	CRANE
	Tom Conrad	NUSC
	Linda Elderhorst	NATC
	Karen Gordon	IDA
	Daniel Green	NSWC
	Steve Howell	NSWC
	Phil Hwang	NSWC
	Daniel Juttelstad	NUSC
	Leigh Lieberman	NADC
	Larry Lindley	NAC
	Warren Loper	NOSC
(Co-Chair OSSWG)	Tricia Oberndorf	NADC
	James Oblinger	NUSC
	Carl Schmiedekamp	NADC

HANDOUTS

890001	Agenda - NGCR Operating System (OS). In-house meeting 18-19 Jan 1989.
890002	NGCR OS Working Group (WG) Directory of Personnel
890003	NGCR Program Description - Cdr. R. Barbour
890004	NGCR Status - Presentation to NGCR Program Review - T. Oberndorf
890005	CBD Announcement 12-28-88
890006	Excerpts from DOP concerning NOS
890007	OS Features Matrix 1/17/89
890008	Brief narrative of candidate OSs
890009	Draft CBD announcement of Brief to Industry in March 1989
890010	STRAWMAN Definition of Terms (T. Oberndorf)
890011	Proposed SAFENET Communications Management Definitions (D. Green)
890012	Questions for consideration by the Policy Group

890013	Questions for consideration by the Approach Group
890014	Requirement for NGCR OS (response by Requirement Group)
890015	SAFENET Presentation
890016	NOS Article from Jan 1988 Local Area Network (LAN) Magazine

ACTION ITEMS:

NADC	Contact SPC and NASA, FAA OS and Lawrence Livermore Lab in regard to membership.
ALL	Call for application requirements documents to Carl.
Carl, Rich	Need to submit s/w concerns to the Back Plane Prototype Working Group.
ATWG	Technology Report from "Available Technology" Subgroup for March meeting.
NADC	Address/e-mailing lists for subgroups?
Tom	Abstract OS.
Dan G.	Remind Safenet of security issue.

Tricia began this first meeting of the NGCR Network Operating System (NOS) Working Group. She handed out the Agenda and Directory of Personnel (#890001 and 2).

Tricia would like points of contact for any other Navy Labs and industry personnel who might be interested in contributing to this Working Group.

Objectives of this meeting:

1. Goal of OS for NGCR.
2. Need Technologies Report before first brief to industry.
3. Agree on Subgroup structure.

Cmdr Barbour presented the NGCR Program Brief (He handed out

NGCR Program Description, #890003).

- Discussed the command structure at SPAWAR.
- Numerous vacancies due to funding limitations.
- Program goals are to increase operational readiness and Program Manager's flexibility.
- Many technology influences. Not all are applicable to NGCR.
- NGCR is an architecture, not a design or a computer.

multiprocessor \
multisystem interconnect -- main areas for NGCR
s/w standards /

- Economic decision: prototype 3 of the 10 NGCR standards
- Focus areas:

SAFENET \ Prototype
BACKPLANE & Conformance Testing
NOS /

Dan G.

Will use commercial standards and select one and modify where Navy needs dictate. Some mods for SAFENET have been acceptable to joint WG, some not.

- Vendors make prototype and these are evaluated for conformance by Navy Labs and presented to Joint WGs.
- Prototype proves standards (Standards are then used to contract for manufacture of units.)
- Can expect a typical 10 year program schedule for these types of development.

Discussion

- NGCR is requesting a RFP for the prototype by Sep 89
  - Need to discuss in this WG if this is feasible.
- What is transition for NOS from current to total NGCR? - undefined

Dan G.

- SAFENET hopes UYK-43, 44.
- Transition to NGCR is highly dependent on target system. Probably a phased transition of NGCR products.

Ralph

- Future bus has been scheduled, but Backplane may consist of several buses

These other buses would handle:

Error log
Maintenance
Debug

Local Bus tightly coupling a few modules. Talk of:
Signal Processing Bus

Current Program Activities

- New tech work:

ruggedized Milspec
Hank, Jerry and Dave are only ones currently attempting to address this

- Ruggedized is between no testing and full Mil testing. Effort is to firm up what ruggedized means.

- Program Program Plan, Acquisition Plan and T&G Plan are currently under revision. Milestone I is just coming up.

ISSUES

Program Doc - not developing only hardware products

Naval Research Advisory Committee (NRAC)- is critical of current NGCR approach

Joint Industry Avionics Working Group (JIAWG)- doing similar things as NGCR but has a different schedule

Tri-Service Program -

SUMMARY

Summary - hardest nut: - Navy policy to implement

TRICIA presents the slides that she had previously presented to NGCR meeting in December, 1988.

Handed out:

NGCR Status 890004
CBD Announcement 890005
Excerpts from Development Options Paper (DOP) 890006
OS Features Matrix 890007

- Same brief as NGCR Program Review in December
- Tricia had been on program less than month
- Not all potential members have yet participated in Working Group
- NOSWG next meeting (in-house) scheduled for February
- Brief to industry by WG in March
- SEI will not be brought in until March meeting
- Suggested Members:

Software Productivity Consortium (SPC) (Tricia does have contacts)

Lawrence Livermore (Tricia does not have contacts)

- Need to do requirements study
- Need to input S/W concerns to B.P. Prototyping RFP
- June 90 award RFP contract for OS standard Finalized 93. How can this work?
- Evolutionary approach - modify existing standard? Does a standard exist, or do we have to start from scratch?
- Emphasis by NGCR is on adoption of industry standard.
- Should OS be distributed or network - Need to make recommendation to NGCR concern what is feasible for O.S.
- Language binding and/or language independent? Make recommendations.
- CBD announcement has been published; heads-up for March meeting.
  - Tricia passed out a paper containing excerpts from the Decisions Options Paper.
  - Presented six main O.S. requirements from DOP Ada-oriented, Real time, Distributed, Multi-Level Security, Reliable, Heterogenous Processors.

Discussions

- Suggestion to replace 'Real time' with 'Critical time.'
- Government would not be maintainer of O.S. - the implementer would be the maintainer, as in commercial systems.
- The Conformance Group is responsible for showing/demonstrating conformance to the standard.

- Conformance testing is not to be used to prove interoperability (Harry Gold). Conformance tests should lead to interoperability.

- If standard requires interoperability, then passing the tests shows interoperability?

- Would like to avoid situation as in DOS where there is only one implementation - it is a standard but unique interface.

- What is possibility that two implementations which meet the standard interoperate?

- Trial use standard and prototype before final standard publication in order to prove standard.

Back to OS Requirements:

- Concept where S/W modules such as schedulers which could be changed based on Achitectures.

- Don't know of an existing standard for an OS.

- Posix is the only known standardization effort, but they have organized into separate WG for Ada, RT, etc.

- Standard Electronics Modules Program (SEMP) CRANE goal would be to provide complete interchangeability over time.

H/W / S/W Context

Ralph :

Currently proposed low performance Backplane has capabilities equivalent to the High Performance backplane described in the DOP (High Performance Computer).

- Discussion on whether to define Million Instructions Per Second (MIPS). Consensus is that definition would be unproductive.

- Two types of Signal Processors NGCR and non-NGCR what does the NOS cover? Does OS Reside on non-NGCR processor or just communicate with these non-NGCR processors.

- Provide paper addressing each of the items contained in the OS requirements from the DOP paper 890006.

Presentation by Dan Green

Handed out SAFENET Presentation #890015

- Presentation originally made to SAFENET Comm Mgt WG
- SAFENET choose to base its work on MAP and then made modifications
- Presented SAFENET Diagram showing the seven ISO/OSI levels and its relation to services, resources, and application
- Same as above for GOSIP

Discussion

- What should the goals of an OS for NGCR be?
- What is the effect on a PM?
- Use multi-vendors product with only a re-compilation
- Interoperability with other conforming components?

(from other vendors)

(ease problems of interop)

PM Advantages:

- S/W transportability
  Interoperability:
  between same processor
  between different board same node
  between different nodes
  between different nodes different vendor
- Purchase of state-of-the-art heterogeneous processors which conform to standard (type of transportability)

0-08

- Maintainability and logistic support

- Raise level of abstraction of design to run on H/W

- Very High Scale Integrated Circuit (VHSIC) OS engine on Silicon

- S/W bases - interface bus

- Other constraints may overwhelm the design, other than OS constraints, such as weight, size.

DBMS Interface

- If we standardized on an Structured Query Language (SQL) DB system, does that mean when someone comes along with an object oriented data base (OODB), it will be non-compliant?

- Device Drivers other than Graphics, will they be subsumed by NOS?

- NRAC committee has some concerns that we need to address.

- Need to provide goal or direction to industry so they can use their initiative to solve problems instead of having them work ad hoc then seeing if we need the technology.

- High need for generating a set of system requirements (specifically RT).

DAY 2

On the second day of the NGCR Meeting, an updated master mailing list was distributed to the attendees. The working group will make heavy use of ARPA net. Also handed out was the STRAWMAN definition for NOS terms, SAFENET definitions (from Dan Green, and an article about NOS terms (#890010, 11, 16).

An attempt was made to define terms, and a suggestion was made to add the following terms:

Node

Needs Modifier - i.e. application

Application Level Interfaces

Interface - One layer to the next layer

Services - Function to be performed

Protocol - Controls transmission of information between two layers

Agree that a computer is based on a Backplane.

- Need to develop common set of terms for use by this WG whether or not it is accepted by other committees.
- Message is very fuzzily defined terms by everyone. ISO uses Protocol Data Unit (PDU) which defines a message between the entities of each layer. A PDU for a layer consists of its Service Data Unit (SDU), which consists of the next highest layer's PDU, plus a header that contains control information for that layer's protocol.
- Need to describe where boundary is between OS and SAFENET.
- NOS could potentially control all levels.
- Need to know computer system architecture or develop a minimal set of hardware.

Karen

- Mach (DARPA) UNIX selected as application interface and extended for specific architectures.
- Extensions for Distributed, secure.
- Executive would be the core of the system. Layered on this would be file system, supervisor, etc., which then form an O.S.

Larry

Can slice horizontally or vertically

Could slice File Sys, Memory Management, I/O, etc.

or

Single Process, Dist. tightly coupled, etc.

Make List

I/O \ Channel Mgt
/ device control

Process Mgt
Interrupt Mgt
Scheduling
Memory Mgt
Interprocess Comm
Information Mgt (including naming)
Resource Mgt
Dispatch Mgt
Time Services (incl Synch)
Fault Tolerance
Performance Monitoring
Debug
Security
Comm Services (math, data conversions, etc.)
Logging/accounting
Recovery/initialization

**Action Put together Glossary of Terms NADC

Tricia takes over to break the Group into four subgroups

Four Groups:

Requirements

Available Tech (use matrix handed out previously)

Approach (standard, family of O.S., etc.)

Policy /legal issues, mandates

Handed out questions for consideration for Approach and Policy (890012 and 890013)

The excerpts from DOP requirements (Requirements Group) and the Matrix (Avail Tech) was handed out yesterday.

Reliability = handling of fault tolerance

Approach WG
---

1. Describe abstract notion of O.S.

2. Address issue of family

   - levels of partition.

3. Describe functionality.

4. Requirements Group would quantify.

5. Available technology Group would provide candidates.

6. Will look at concepts and generate capabilities w/o regard to current technology.

7. Will not come out with something as formal as CAIS.

8. Assume Ada

Dan G:
---

9. We should expect a 4-10 year gestation.

Tricia:

10. Need spec by mid 1990's to be used by projects to :

   - specify for their systems.
   - be used by real projects.

Linda:

11. JIAWG is working on near term problem CORE-SEE while working on the long term (full) SEE. This could be method to evolve to full system.

Rick:
---

12. NGCR has no plans for this type of activity (evolution).

Tricia:
---

13. Our spec in '93 will have to be close to final. In order to have a viable prototype, we need a draft specification by '91.

14. Difference between Standard and Spec:

Spec - describe what to build
Standard - describe how to build.

Discussion

Chances are that, if we solve all current problems, we will fall short for systems 10 years from now. We will have to attempt to look down the road.

(Dan) argues for evolutionary approach, see what's broken and fix it, rather than starting with blank sheet of paper then moving to conceptual design.

Technology Awareness Group

Need "Technology Report" for March meeting describing the available technology - like NOSC White Paper (short excerpts).

SAFENET

- Need to define boundary between OS and SAFENET.
- If OS has hooks and Netw has hook they need to match.
- Avoid redundantly specifying functions/features in OS and SAFENET (particularly specify differently).
- Security.

Planning for March Meeting

March 16 and 17

Reworked CBD announcement

Agenda for meetings
- Updated Matrix and Technology Report
- Presentation on what discussions have been

February Meeting (In-house Working Group)

Next Meeting: February 21, 22, to start at 10:30 on the 21st...

LIST OF ACRONYMS

ATWG	Available Technology Working Group
DOP	Development Options Paper
FAA-OS	Federal Aviation Administration - Operating System
JIAWG	Joint Industry Avionics Working Group
LAN	Local Area Network
MIPS	Million Instructions Per Second
NGCR	Next Generation Computer Resources
NOS	Network Operating System
NRAC	Naval Research Advisory Committee
OODB	Object Oriented Data Base
SPC	Software Productivity Consortium
SQL	Structured Query Language
VHSIC	Very High Scale Integrated Circuit
WG	Working Group

NEXT GENERATION COMPUTER RESOURCES PROGRAM

OPERATING SYSTEMS STANDARDS WORKING GROUP

INITIAL MEETING 16-17 MARCH 1989

MEETING MINUTES

The Next Generation Computer Resources (NGCR) Program Operating Systems Standards Working Group (OSSWG) initial meeting with industry was held March 16-17 at the Naval Surface Warfare Center (NSWC) in White Oak, Maryland. Over 200 representatives of government, academia, and industry gathered to hear NGCR presentations on Thursday morning. The group was then organized into three subgroups that met Thursday afternoon and Friday morning, with subgroup wrap-ups presented and the meeting concluded Friday afternoon.

Thursday, 16 March.

The meeting was opened with welcome presentations from CDR Rick Barbour, SPAWAR 324A, and Captain Robert P. Fuscaldo, Commanding Officer, NSWC-White Oak. All presentations were filmed, and a limited number of copies will be made available to those who wish to make a copy of their own.

The Keynote Address was given by RADM R. L. Topping, Director, Warfare Systems Engineering, SPAWAR. ADM Topping stated that the current joint NGCR cooperation between the Navy and private industry is a "superlative" success, as demonstrated by the SAFENET and Backplane Bus working groups. His presentation then addressed the importance of computers to Battle Force Systems Engineering. Noting that weapons capabilities, the warfare domain, battleground distance, sensor capabilities and C^3 tactics have all increased exponentially over the past 200 years, ADM Topping explained how the Navy wants to apply systems engineering toward programs to structure today's battle force. He identified three major challenges to battle force programs: allocation of functions to resources (platforms); intra/interoperability across major platforms (ships, submarines, aircraft, satellites); and weighted trade-off of all system capabilities and engineering disciplines. ADM Topping then pointed out that many of the battle force pervasive disciplines apply to "information warfare". These include interoperability (standard interfaces), graphics (interaction with human factors), computer standards, multi-level security, and data base management systems. He concluded by saying that the Navy was structured to meet these challenges in the 1990's with the joint industry/government venture for achieving common interface standards for computer resources (SAFENET and Backplane Bus) and the OSSWG providing the "glue" to tie the intra/interfaces together.

A copy of ADM Topping's presentation is available upon request.

No questions were asked of ADM Topping.

The next presentation was made by Hank Mendenhall, NGCR Program Manager, SPAWAR 3243. Hank identified the NGCR Program purpose, organization, history of developments (since December 1985), statement of needs, and program objectives. He stressed that an open systems architecture approach is what the Navy

needs, and the joint industry/Navy standards program is the solution. The advantages to the Navy in using an NGCR open systems architecture approach were then identified by Hank. These included: rapid incorporation of state-of-the art technology into the fleet, logistics reduction due to commonality of NGCR products, interoperability of NGCR products through common interfaces, program manager flexibility in system design and acquisition managament, and the establishment of a commercial base of standards and products which will provide industry with incentive to invest in the NGCR program. Several times throughout the presentation, Hank stressed that NGCR is a significant departure from past Navy policies involving computer resource standardization. He stated that NGCR is not a computer or software system, but, instead, an architecture that standardizes hardware/software interfaces and protocols. He reminded the audience that the parameters of the open system architecture would be defined by joint industry/Navy working groups, focusing on widely used non-proprietary commercial standards. The NGCR standardization areas were identified as multiprocessor interconnects, multisystem interconnects, and software entities such as operating systems, database management systems, programming support enviroments, and a graphics language/interface. Further elements of the NGCR program, such as program elements, flow, and milestones were discussed as well as the working group interrelationships and current program activities.

A copy of Hank Mendenhall's presentation is available upon request.

Several questions were asked of Hank. The following are edited representations of the questions and answers:

Q: Is the purpose of today's meeting to define hardware and software interfaces between different types of computer systems like Apple MacIntoshs and IBM PCs?

A: That is a defintion the Navy frequently uses for open systems architecture - different types of computers talking over standard interfaces.

Q: Please repeat your comment on the use of proprietary standards.

A: The standards that we will publish will not be proprietary.

Q: What is the Navy's position on NATO involvement in the NGCR Program?

A: All foreign governments are invited to participate, and foreign companies are currently represented in the NGCR working groups.

CDR Rick Barbour, NGCR OSSWG Chairman, SPAWAR 324A, then presented an overview of the NGCR OSSWG. He presented the purpose and approach of the OSSWG, then gave the proposed charter. Consistent with the NGCR Program approach presented by Hank Mendenhall, CDR Barbour reiterated that the Navy would like to initiate a dialogue with industry in order to form a joint industry/Navy operating systems standards working group. This group will establish NGCR operating system interface standards which will be based on the "best fit" between Navy requirements and existing commercial operating system standards. The OSSWG will meet every 6 weeks and plans to produce an operating system requirements statement and draft abstract model by July '89, a technology report by October '89, and a product evaluation process by January '90. The OSSWG will select concepts/products to be incorporated into a draft baseline standard by September 1991. CDR Barbour acknowledged that this is an ambitous schedule and that the working group was faced with many challenges.

A copy of CDR Barbour's presentation is available upon request.

Several questions were asked of CDR Barbour. The following are edited representations of the questions and answers:

Q: Is there a mandate to develop this operating system in Ada?

A: Mandate is a strong word. Since DoD requires new software development in Ada, it is something we must consider.

Q: Academia seems to be underrepresented in this (NGCR) effort.

A: We need to look at what is being done in the academic arena.

Q: Isn't it true that DoD directives emphasize the use of existing (software) products whether they are written in Ada or <u>not</u>?

A: That is correct. We will be examining the incorporation of existing commercial products whether or not they are written in Ada.

Tricia Oberndorf, Co-Chairman OSSWG, NADC Code 7031, then gave a presentation expanding on CDR Barbour's. Tricia first defined several high level requirements for the NGCR operating system. These were: real-time, distributed, Ada-oriented, heterogeneous, mulit-level secure, and fault tolerant. She revisited the Ada-orientation requirement and offered that this was a reference to Ada Languange bindings; it did not mean that implementations of the operating system had to be written in Ada. She then went on to say that the peculiarities of each requirement would tend to pull the focus of the OSSWG in different directions and that compromises would have to be made

in order to adequately address the Navy's needs. Tricia then reviewed a series of questions addressing the relevant issues before the OSSWG. These included the orientation of the system that the Navy required to have an interface standard (operating system, executive, kernal, Ada run-time), the language dependencies of the standard, and the relationship of the operating system standard with other NGCR standards. Tricia's list of issues also touched upon performance characteristics and the question of Navy needs being met by existing technology. Tricia then identified the proposed subgroup organization of the OSSWG, consisting of a Requirements Subgroup - chaired by Rich Bergman of NOSC, an Available Technology Subgroup - chaired by Phil Hwang of NSWC (White Oak), and an Approach Subgroup - chaired by Tom Conrad of NUSC (Newport).

A copy of Tricia Oberndorf's presentation is available upon request.

Each of the Subgroup chairmen gave a presentation describing the objectives, approach, products, and milestones of their respective group. The following paragraphs are a synopsis of those presentations.

The Requirements Subgroup will identify the requirements for the NGCR operating systems and support the development of a set of operating systems interface standards. They will examine current Navy and commercial operating system functions, performance requirements, hardware/software/language interface issues, various "-ilities" issues (security, technology insertion, etc.), and then produce a requirements document. The schedule calls for a Requirements Statement on 6/1/89, an Operational Concepts Document Draft on 1/1/90, and an Operational Concepts Document Final on 5/1/90.

The Available Technology Subgroup will examine the incorporation of current and evolving operating systems interfaces, services, and protocols into the NGCR operating systems standards. In addition to identifying issues associated with existing and evolving operating systems functions and interfaces, they will develop a representative model of existing operating systems architectures. This Subgroup will also evaluate existing and evolving technologies relative to the criteria for NGCR technologies. This Subgroup will consolidate comments from commercial developers by 4/16/89 and produce an operating systems technology report by 10/1/89. An evaluation report summarizing technology evaluation results is due 4/1/90.

The Approach Subgroup will define a process for establishing a family of NGCR operating systems interface standards. This activity includes identifying the process for industry/Navy cooperation in defining standard operating systems interfaces, services, and protocols. This group will also define the procedures for comparing identified operating systems elements against identified NGCR requirements in addition to defining appropriate OSSWG documents. The Approach Subgroup is scheduled

to deliver a Requirements Document DID on 6/1/89, an abstract model of operating system services on 7/1/89, and the Evaluation Process Definition Draft on 9/1/89 and Final on 1/1/90.

A copy of the presentations for each of the Subgroups is available upon request.

After the Subgroup presentations, there was a general question and answer period hosted by CDR Barbour and Tricia Oberndorf. The following are edited representations of the questions and answers:

Q: Who is leading the OSSWG effort?

A: The NGCR Program Office.

Q: Where will future OSSWG meetings be held?

A: That is up to the individual subgroups. General sessions of the entire OSSWG will be held in the Washington D.C. area for the next 4-5 meetings.

Q: Is it the Navy's intent to create an IEEE standard?

A: No, that will be up to industry and the IEEE.

Q: What is my incentive to pursue this Navy market?

A: Hopefully, the market created by the joint industry/Navy standards will be expanded to the commercial market.

Q: Why should we voluntarily participate in this working group? What is in it for us to attend? It seems that a more prudent approach would be for us to stand by and watch what developments occur.

A: (At this point, John Machado was introduced to the group to field the question.) Many of you in attendance currently are involved with building systems for the Navy, and you know what current Navy requirements are. If you are actively involved with guiding the requirements for NGCR, you will be very familiar with future Navy and possibly commercial systems requirements. Your familiarity with these new standards will add to your competitive abilities.

Q: That is an explanation for why we should watch, not for why we should participate.

A1: Without participants from industry, the Navy is not sure we will ever meet NGCR objectives.

A2: (Dan Green, NSWC-Dahlgren, and also a member of the SAFENET working group, provided additional comments.) There are other reasons to participate. These working groups provide a learning experience for your people. It gives them a chance to work side by side with other people who are sometimes competitors and who are involved with the same type of products. It is also your opportunity to be involved with the leading edge of industrial and Navy technology and get in on the ground floor of defining that technology.

Q: What interest has the Army and Air Force expressed in NGCR?

A: There is currently no DoD planning to make NGCR a joint service effort. The Army has expressed some interest in NGCR and the Air Force is aware of the program. Both services are watching closely.

Q: Has this effort been scoped to the point where, for example, there are requirements for a particular platform?

A: We are currently examining <u>all</u> requirements and need more input. It is a difficult job.

At this point in the meeting, CDR Barbour asked the attendees to chose a particular subgroup in which they would like to participate. Each Subgroup received approximately the same amount of interested parties. For the remainder of the day and for half of Friday morning, the Subgroups held open discussions among their members and attempted to identify issues and assign them to focus groups created within each Subgroup.

<u>Friday, 17 March</u>

Subgroups reconvene.

Most of the morning session was a continuation of subgroup meetings from the previous afternoon. At the conclusion of the subgroup meetings, a brief presentation* was given by each of the subgroup chairmen.

*NOTE: The "raw" viewgraphs presented by each of the subgroups <u>is</u> included as an attachment to these minutes.

A "wrap-up" presentation was then made by Tricia Oberndorf. She concluded:

- The schedule for the OSSWG is ambitious. It also is the schedule! We need to keep pace with the SAFENET and Backplane Bus working groups.

- There is a recognized need for support from industry, but this working group support will not be funded by the NGCR Program Office. This is a cooperative endeavor.

- The use of E-mail/ARPANET is strongly encouraged for working group participants. Each Subgroup chairman's account number is given in the handout and there will be additional accounts set up on a Subgroup basis.

- Comments are requested on the "NGCR OSSWG Procedures" document contained in the briefing handout package (near the end).

- Coordination among the three Subgroups is essential and will be accomplished via E-mail, Subgroup schedules and deliverable reviews, executive committee meetings, and the meetings of the entire OSSWG which take place every 6 weeks.

- Coordination among the various NGCR working groups (SAFENET, Backplane Bus, Conformance Test Committee, etc.) will be established through the NGCR Program Office.

- Document distribution among the Subgroups will take place through the Subgroup chairmen.

- Administrative support for the OSSWG will be arranged through the NGCR Program Office.

- Several copies of the video tape of the meeting may be made available to be copied by interested parties. Future OSSWG meetings at NSWC - White Oak would also be video taped.

- Future OSSWG meetings may be extended to 2 1/2 days (from the current 1 1/2).

Tricia also stated that the objectives for the first OSSWG meeting have been met. They were:

- OSSWG participants got to know each other.

- The OSSWG executive committee received confirmation through working group participation that the original plans and ideas for the OSSWG make sense.

- Progress was made, the OSSWG is off to a good start.

- There have been presentations arranged for the next meeting:
  -- A presentation on POSIX
  -- A presentation on DARPA projects.

A "wrap-up" presentation was also made by CDR Barbour. He reiterated the Navy's desire to initiate a dialogue with industry to:

- form a joint industry/Navy working group
- adopt industry standards (if feasible)
- take advantage of the best available technology in industry
- ensure the availability within the industrial base of a family of non-proprietary operating systems standards
- RAPIDLY FIELD TODAY'S TECHNOLOGY TO THE FLEET!

Schedule for future OSSWG meetings

The initial OSSWG meetings will take place in the Washington D.C. area. Future meetings may be at other locations.

1989 DATES
May 16-18*
June 20-21
August 1-2
September 12-13
October 17-18
December 5-6

The next meeting (16-18* May) will be held at NSWC - White Oak, Maryland with Registration at 0800 and the meeting start at 0900.

*NOTE: change in original schedule for May meeting

NEXT GENERATION COMPUTER RESOURCES PROGRAM

OPERATING SYSTEMS STANDARDS WORKING GROUP

MEETING 16-18 MAY 1989

MEETING MINUTES

The Next Generation Computer Resources (NGCR) Program Operating Systems Standards Working Group (OSSWG) second meeting was held May 16-18 at the Naval Surface Warfare Center (NSWC) in White Oak, Maryland. Approximately 100 representatives of government, academia, and industry gathered to hear presentations given to the plenary session on Tuesday morning. The group then disseminated to the three on-going subgroups, with working meetings taking place the remainder of Tuesday, all day Wednesday, and Thursday morning. Subgroup wrap-ups were presented and the meeting concluded Thursday afternoon.

Tuesday, 16 May.

CDR Rick Barbour, SPAWAR 324A, opened the meeting by reiterating the charter and objectives of the OSSWG, and then turned the floor over to the subgroup chairmen for brief status briefings/reports.

Rich Bergman, the Requirements Subgroup chairman, stated that the subgroup was continuing to identify the requirements for the NGCR operating systems interfaces. At this meeting, the focus groups would be formalized, the OS architecture model would be reviewed, and an initial set of requirements for the Requirements and Operational Concepts documents would be formalized. Rich also stated that the focus group members would be assigned tasks to complete prior to the June OSSWG meeting.

Phil Hwang, chairman of the Available Technology Subgroup, discussed several products of the subgroup, identified seven focus groups, and outlined the activities and schedule for deliverables. The subgroup planned to produce the Operating Systems Technoloy Report first draft by the end of this meeting (or shortly after) and would also attempt to identify a reasonable set of Technology Survey issues for the June meeting. Phil also identified plans for a summer OS technology workshop and the development of a pool of real-time operating systems to survey and evaluate. The Subgroup had also scheduled two speakers, James F. Ready (Ready Systems) and Marvin Shugarman (BiiN Federal Systems), to make presentations to the Subgroup Tuesday afternoon.

Tom Conrad, chairman for the Approach Subgroup, identified the current Subgroup participants and product milestones. The POA&M for the Subgroup had been completed, the Abstract Model would be available 7/89, the DID for the OCD would be completed 6/89, and the Evaluation Process Document draft was scheduled for 9/1/89 with the final document due 1/1/90. Tom identified the main plans for this meeting as completing a complete OCD DID draft for review, discussing and revising the Abstract Model, and devising a plan of action for the Evaluation Process definition.

Consistent with one of the March meeting "wrap-up" comments, the OSSWG has begun to coordinate its activities among the various NGCR working groups. The next three presentations were made by members of two of the other NGCR working groups

representing SAFENET and the Backplane Bus efforts.

Dan Green (NSWC-Dahlgren) presented a briefing on SAFENET. He noted that efforts were underway to complete the SAFENET I standard and handbook, continue SAFENET II development, and begin to develop user guides. Dan then identified key members of the various SAFENET subgroups and the executive committee. To give the audience a sense of the extent of SAFENET implementations, Dan provided a partial list of "declared", "probable", and "potential" SAFENET users. Because of Navy-unique requirements and various shortcomings in commercial LAN standards, the working group decided against adoption of a single existing standard for SAFENET. The presentation portrayed SAFENET I as a profile/combination of ISO, IEEE, and MAP protocols. Dan also indicated that the working group was looking for help in defining SAFENET "lightweight" protocols (SLWP), and that they were currently working with the XTP design team and X3S3.3 committee to resolve these. In closing, Dan identified several issues for OSSWG consideration. He inquired as to the best method to interface SAFENET to the NGCR OS, such as at the top of the "transport" layer. Additionally, he questioned how to determine data transfer requirements for the interface and suggested that special services of the data transfer system be identified.

The next speaker was LCDR Harrison Beasley, SPAWAR 324, who gave a brief overview of the NGCR Backplane Bus. LCDR Beasley informed the group about the selection of the Futurebus Standard for NGCR and reviewed the current status and schedule.

Dwight Wilcox of NOSC gave a more technically detailed Futurebus presentation, specifically on the "Peri dic Phase Adjustment Method of Distributed Clock Synchronization". The presentation examined the potential applications, advantages, and methods for synchronizing distributed clocks located on individual interface cards in a distributed backplane bus architecture. Dwight then gave instructions as to adjusting and building adjustable rate clocks. He also touched upon the concept of clock synchronization (via token passing) in a local area network environment.

The final two presentations of the morning session addressed the state-of-the-art in operating systems technology, POSIX and Mach.

Dr. Douglass Locke, IBM Systems Integration Division, briefed the group on "POSIX in Realtime Systems". Dr. Locke identified POSIX as the "definition of an operating system interface". He noted that: the Basic POSIX Interface Standard (IEEE 1003.1) was completed in August, 1988; the Shells and Utilities Standard (IEEE 1003.2) was now in the IEEE balloting process; the Real-Time Standard (IEEE 1003.4) was scheduled for completion in June, 1990; and the Ada Bindings Standard (IEEE 1003.5) would be finished by March, 1990. No schedules were given for the Secure/Trusted Standard (IEEE 1003.6), the Network Interfaces Standard, or the Transaction Processing Standard.

Dr. Locke reviewed the POSIX standard functions, indicating that they were very similar to, and modelled after, UNIX operating system functions. In fact, like UNIX, POSIX is written in the C programming language.

The remainder of Dr. Locke's presentation focused on real-time extensions for POSIX. The topics covered were priority scheduling, asynchronous event notification, timers, shared memory, real-time files, semaphores, IPC message passing, process memory locking, asynchronous I/O, and synchronous I/O.

The final presentation of the Tuesday plenary session was given by Dr. William Scherlis, Program Manager of Computer Systems, DARPA Information Science and Technology Office (ISTO). Dr. Scherlis briefed the group on the new, high performance, portable operating system base, Mach.

Dr. Scherlis detailed the DARPA funded evolution of Berkeley UNIX from AT&T UNIX, highlighting the new features such as virtual memory, extended process control, a new file system, and advanced network support. He explained that the project to provide BSD UNIX with major innovations has nearly reached its logical end, and the critical research items that had been developed (the new file system, application interfaces, OSI interfaces, and POSIX) will be transferred to Mach. Dr. Scherlis stated that Mach (completely new) will eventually replace Berkeley UNIX as the base operating system in the U.S. R&D arena, and that Mach will be object code compatible with Berkeley UNIX. He also identified an NSA "Orange Book" compliant, multi-level secure (unclassified to Top Secret) version of Mach, TMach, that is scheduled to become available in FY 92.

In addition to enhanced OS performance, a strong feature of Mach is its portability to a variety of architectures, accomplished through extensibility of the basic Mach system and emulation of other operating systems.

To conclude his briefing, Dr. Scherlis provided a list of Mach users and a schedule of DARPA activities for distributed systems in the upcoming fiscal years. These included meeting requirements for Mach support, security, real-time enhancements, and suitability for military operating systems.

The remainder of the day consisted of Subgroup working meetings.

Wednesday, 17 May

Subgroups reconvene.

Thursday, 18 May

Subgroups reconvene.

Most of the morning session was a continuation of Subgroup

meeting wrap-ups, with each individual focus group (within each Subgroup) compiling several viewgraphs illustrating their respective charters and planned activities. The "lead" person of each focus group then presented the viewgraphs to the reassembled OSSWG. The "raw" viewgraphs are included as an attachment to these minutes. The following paragraphs do not provide the detail of each focus group, but, instead, are oriented to reflect the results on a Subgroup level.

Requirements Subgroup

The Requirements Subgroup activities for this meeting included reviewing their charter, reviewing pertinent documentation (the Abstract Model draft, Implementation Sample, and March meeting information), and formalizing five focus groups. These are the Execution Model, Fault Tolerance, Distribution/Real-Time, Security, and Management Issues focus groups. The Subgroup produced a set of requirements for the Initial Requirements Document and OCD, established a schedule, and made individual assignments to its members.

For the June OSSWG meetings, the Requirements Subgroup will develop the first draft of Execution Model requirements, and, in early July, develop the first draft of the OCD. The first revision of the OCD will be produced and reviewed during the September meeting and presented to the OSSWG for the October meeting. After incorporation of comments, the final OCD will be reviewed during the December OSSWG sessions, then released for broader review. The final OCD (for publication) will be released next May.

Available Technology Subgroup

The Available Technology Subgroup had already organized into seven separate focus groups at the March meeting. These were the Architectures, Software, Fault-Tolerance, Real-Time, Framework, Standards, and Security focus groups. During this gathering, the Subgroup further explored their issues list to lower levels, produced a draft of the Technology Report, initiated dialogue with other Subgroups, and assigned tasks to its members.

Future plans for the Subgroup include a revision of the Technology Report for the June OSSWG meeting, a survey of real-time operating systems to evaluate, development of an initial matrix of operating systems and characteristics, and an examination of evaluation methodology and critera.

Approach Subgroup

This Subgroup consists of the OCD DID, Abstract Model, and Evaluation focus groups. During this set of meetings, the Approach Subgroup completed a walk-through of the draft OCD DID and Abstract Model, established a new baseline of documents, developed a plan for generating an evaluation process report, and met with other Subgroups.

The schedule for the OCD DID is as follows: Version 1.0, 5/25/89; Version 1.1, 6/16/89; briefing to OSSWG, 6/20/89.

Closing Comments

The next OSSWG meeting will take place June 20-22 in Crystal City (Arlington), Virginia. The general opening session on the 20th and closing session on the 22nd will be at Vitro Corp. and the break-out sessions will be held at Booz, Allen and Hamilton Inc. Details are included as attachments to these minutes.

The tentative plenary session presenters for the June meeting are Mike Kamrad of UNISYS Computer System Division, Roger Martin of the National Institute for Standards and Technology (NIST, formerly NBS), and members of the Approach Subgroup. Mr. Kamrad will give a presentation on the ARTEWG (Ada Run-Time Environment Working Group), Mr. Martin will speak about NIST software standardization work, and the Approach Subgroup will present their results on the OCD DID and abstract model of operating systems services.

Tricia Oberndorf once again stressed the importance of the use of E-mail by OSSWG participants. It is the most effective and efficient way to coordinate OSSWG documentation and messages. For information on the NADC.ARPA machine, contact Carl Schmiedekamp (NADC) at (215) 441-1779.

As the meeting came to an end, Tricia requested that the entire OSSWG take on one additional responsibility for the June meeting: be prepared to discuss and identify what we (the OSSWG) need to standardize, such as operating system kernel functions and the Ada run-time environment features.

CDR Barbour presented the closing address to the group, congratulating the members on the progress that the group has made and encouraging active future participation in the OSSWG.

NEXT GENERATION COMPUTER RESOURCES PROGRAM

OPERATING SYSTEMS STANDARDS WORKING GROUP

MEETING 20-22 JUNE 1989

MEETING MINUTES

The Next Generation Computer Resources (NGCR) Program Operating Systems Standards Working Group (OSSWG) third meeting was held June 20-22 at the Crystal City (Arlington) offices of Vitro Corp. and Booz-Allen & Hamilton, Inc. Approximately 80 representatives of Government, industry and academia gathered to hear presentations given to the plenary group all day Tuesday. Wednesday and Thursday were allocated to Subgroup working meetings with wrap-up presentations given late Thursday morning.

Tuesday, 20 June.

CDR Rick Barbour, SPAWAR 324A, once again gave the welcome presentation to the group, noting that the OSSWG was approaching several important July milestones for the Operational Concept Document (OCD), the Abstract Model, and the Requirements Statement. CDR Barbour stated that this meeting's objectives would be to establish a consensus on the scope of the Operating Systems Interface Standards. He also expressed a need for volunteers to serve as OSSWG meeting hosts during 1990. The floor was then turned over to the Subgroup chairmen for status reports.

Rich Bergman, Requirements Subgroup chair, outlined the Subgroup's progress since the May meeting. They have reviewed the OCD DID, Abstract Model and other relevant information, in addition to continuing to develop OS interface requirements. For this meeting, the group planned to begin fitting requirements to the OCD and to review the Execution Model.

Phil Hwang, chairman of the Available Technologies Subgroup, identified the accomplishments of his group and their current plans. Phil said that they were several levels down in detail on their Technology Survey Issues list and the list had also been expanded. In addition, the Focus Groups have produced draft study inputs concerning their respective areas and continue dialogue with the other subgroups. During this meeting, the group planned to produce a new version of the Technology Report, a survey evaluation of operating systems, and an initial matrix of operating systems and characteristics. The Subgroup also planned to continued to develop the evaluation methodology and criteria during this session. Phil also identified several potential risk areas in meeting document milestones. These included Sub/Focus group participation, the short time frame and the lack of consensus on the scope of deliverables and the OSSWG charter.

Tom Conrad, of the Approach Subgroup, also reviewed the products and plans for his group. The OCD DID first draft was released in May, with draft 1.1 scheduled for distribution 16 June. The Abstract Model draft 0.9 was available and would be presented to the OSSWG later Tuesday morning. Group comments to the model would be incorporated into draft 0.99, 27 June, and an Initial Release, 1.0 was planned for 30 June. Tom stated that the group was also discussing approaches to the evaluation process and was beginning work on a draft of the Evaluation Process Document.

The next presentation was made by Roger Martin of the National Institute of Standards and Technology (NIST), formerly the National Bureau of Standards (NBS). Roger began the presentation by stating a NIST objective to "provide (a) vendor independent way for federal agencies to specify operating system environment requirements which promote applications portability." The presentation examined the use of the POSIX operating system interface standards to achieve the NIST goal. It was noted by Mr. Martin that POSIX alone will not be sufficient to achieve portability for all applications, but, instead, there is a need for an open systems architectural approach to applications portability. The many benefits of an open systems architecture were presented, followed by an architectural model called the "applications portability profile (APP)." The Federal strategy for the APP is to evolve it as an open process, leveraging existing standards where possible and initiating new standards development where needed. The strategy includes the development of collaborative partnerships among government, public and private sector participants as well as formal working groups in the operating systems community. These partnerships will build a consensus on open systems, defining functional characteristics, selecting non-proprietary standards, soliciting commitment from open systems architecture product vendors and users, and developing conformance tests (Editors note: similar to NGCR strategy). Mr. Martin concluded his presentation with a history of POSIX followed by a detailed accounting of all IEEE P1003.X (POSIX related) standards development activities. He stated that FIPS has adopted draft 12 of P1003.1, and will eventually recognize the current draft, 13. There are 23 modifications of IEEE P1003.1 in FIPS 151, including correction of technical errors and setting mandatory requirements in 151 for options and behavioral alternatives allowed by P1003.1 where portability would be affected.

A briefing on the Ada Runtime Environment Working Group (ARTEWG) was then given by Mike Kamrad of Unisys. Mike began by reviewing the current concerns of the Ada community. These ranged from the shortcomings of Ada to the lack of knowledge of Ada users. He then gave the charter of the ARTEWG, a SIGAda sponsored group of Ada users and implementors. The ARTEWG establishes conventions, criteria, and guidelines for Ada runtime environments. It also provides a mechanism for interface between Ada users and implementors. The ARTEWG's initial plan of action will be accomplished over the next two years, producing the following baseline products: a catalog of implementation dependencies for runtime environments, guidelines for effectively using Ada runtime environments, and a catalog of Ada runtime environment interface proposals with rationale and feasibility. Mr. Kamrad noted that Ada implementation technology should be further examined. There is a need for more technical and proof of concept developments resulting in more knowledge being obtained about Ada runtime requirements. He then spoke about current

ARTEWG activities, noting the conservative approach that ARTEWG has observed in the Ada 9X program . Mr. Kamrad also gave a graphically-oriented presentation of a compiled Ada program and runtime environment. He illustrated the relationships of the compiled and linked Ada code to the Ada executive in both a hosted OS and "bare bones" machine environment. He pointed out that the efficiency of the generated runtime environment is directly related to the compiler vendor's ability to link the optimal number of Program Library Language routines to code sequences and data objects of the applications program.

Tom Conrad, Approach Subgroup chairman, then gave a presentation on the recently developed OCD DID. Tom outlined the 7 sectionsof the DID, detailing the proposed eventual contents of each section. The sections are the Scope, Applicable Documents, Mission, OSS Functions and Characteristics, Government Agencies, Notes and Appendix.

Dr. Carl Schmiedekamp, NADC, briefed the group on the OSSWG Architectural Model for Embedded Operating Systems, more commonly called the Abstract Model. Dr. Schmiedekamp's discussion approached the model from 4 different views: the Application Domain, Interface, Services, and Process. According to draft version 0.9, "the OSSWG architectural model for embedded systems is a conceptual model which provides a context for the description of application developers' requirements, a context for description and comparison of existing operating systems, and a framework for the specification of Operating Systems Standards (OSS) for embedded systems." The Applications Domain view illustrated the use of an NGCR-compliant OS by a variety of typical applications. The interface view depicted a Local Processor Operating System (LPOS) as the central node to a range of functional interfaces. The services view discussed the operating system in terms of managing explicit and implicit services, both synchronous, such as processing "request" queues, and asynchronous, such as processing interrupts. Dr. Schmiedekamp then introduced the System Resource Allocation Executive (SRAX). The SRAX acts as the single operating system for the whole system (or series of LPOSs), managing resources. The SRAX allows local processes to use remote devices in a seamless manner.

The final presenter on Tuesday was Tricia Oberndorf (NADC) who directed an open forum on "defining the scope" of the NGCR OS Standard. Tricia showed the group an emerging view of the OS. This view depicted the OS as a collaboration of software modules (termed "internal") communicating with the outside world through a set of well defined interfaces. After a series of discussions, it was decided that the precise definition of these interfaces would evolve into the OS standard. The presentation addressed many of the "loose ends" in the OSSWG efforts. Various OS alternatives (such as kernels vs. full operating systems) were examined, compared, and contrasted in an effort to address what (exactly) should be standardized in the OS area to best serve NGCR and

platform applications software. The question of (distributed vs. centralized) functional allocation and control was raised, prompting further discussion of the Abstract Model. The group agreed that no single entity could be labeled the "NGCR OS," and that a more appropriate approach would be a family of cohesive interface definitions, selectable to meet the needs of the particular application. Tricia led the group in addressing 7 issues that she had previously developed. After several hours of deliberation, all the issues were resolved. The results are as follows:

Issue	Resolution
Interface Aspects	1) Consider both interfaces for applications programs and other interfaces as necessary. 2) The OSSWG should not be concerned with the internal (inter-module) interfaces of the operating system. We should only define external interfaces.
Specification vs. Implementation	The OS specification may specify an entire set of services. An implementation may invoke a sub- or superset of the interfaces in the specification.
Layering	Layering is implementation specific and not of concern to the specification.
Configurability	Configurability, or, more appropriately, tailorability, is desirable in the standard.
"Native" OS Presence	The presence of a native OS or implementation on a "bare bones" machine are of no consequence to the standard.
Languages other than Ada	Ada should be the only language used in developing the OS standards, but, realistically, the C language should be considered. (POSIX is in C)
"Family" - Meaning - Nature of Family Relationships	The OS "family" will most probably be a kernel with selectable additional services.

Wednesday, 21 June

Subgroups convene.

Thursday, 22 June

The Subgroups reconvened Thursday morning from 8:00 am to 10:30 am. After the Subgroup meetings adjourned, the working group as a whole reconvened to hear Subgroup reports and wrap ups from CDR Barbour and Tricia Oberndorf.

Approach Subgroup

Dr. Schmiedekamp presented the report for the Approach Subgroup. He summarized their accomplishments for this meeting, which included a final edit of the OCD DID, a draft outline for the evaluation process report and significant interaction with the Requirements and Approach Subgroups to coordinate their efforts. There were a number of additions to the OCD DID to reflect working group consensus on performance levels and to bring sections of the OCD into agreement with the Abstract Model. The Subgroup did a walk-through of the Abstract Model, produced a minor revision (0.91) and identified the need for a more significant update to move the interface discussion from the appendix into the main text, and to incorporate the "family member" view of operating systems. In addition, the Abstract Model has been renamed the Reference Model. The Subgroup developed an outline of the Target Domains of systems, which produces a matrix of 72 points in the "family space." They are discussing ways to reduce this number to a managable number of families.

The group's evaluation report outline has 5 sections: Introduction, Approach, Evaluation Criteria, Evaluation Process and Results/Summary. The Evaluation Criteria will likely be suplemented with sub-criteria. The group is looking at a matrix approach to evaluation, however the large number of family members could make this a significant effort. Two approaches to weighting have been identified. The first is to develop a set of weights for each feature for each important family member and evaluate each OS against each member. The alternative is to weight each OS feature for each family member, and then weight the importance of the family members to develop an aggregate weighting scale for evaluating candidate operating systems. The Subgroup hopes to reduce the 72-point space to its 6-8 most important members for evaluation. Plans for future work include availability of a final OCD DID and a revised Reference Model via the ArpaNet by 30 June, and, at the next meeting, restructuring the Subgroup to focus on evaluation and to review the OS requirements in relation to the Model.

Available Technology Subgroup

Phil Hwang then gave the report for the Available Technology Subgroup. At this meeting, they generated the next draft of the Technology Issues report, developed a list of operating systems for the survey and assigned each Subgroup member 2-3 OS's to evaluate. Finally, the Subgroup developed a survey form for the evaluation process, following the form of the Reference Model. Surveys are due back from members in 3 weeks. The surveys will be sent to OS vendors for a "sanity check," intended to ensure that the various candidates are accurately represented. At the August meeting the Subgroup expects to have the surveys back from the vendors, and they will produce the next draft of the Technology Issues report.

Phil followed the Subgroup report with the announcement of an Operating Systems workshop, co-sponsored by the Navy, the Institute for Defense Analyses (IDA), and the University of Maryland, to be held 19-20 September at the University. The workshop is intended to provide a forum to consider OS issues for mission critical systems, and an opportunity for vendors to discuss their products' application in that area. Some debate was held regarding the scheduling of the Workshop, and it was agreed that arrangement would be made to videotape the proceedings for those OSSWG members who could not attend.

Requirements Subgroup

Rich Bergman presented the wrap-up report for the Requirements Subgroup. This Subgroup reorganized their Focus groups to continue their work; the new Focus groups are Security, Fault Tolerance/Performance Monitoring, Multi-Languages Interface, External Interfaces, and Real Time/Distributed Processing. Each Focus group was assigned a set of services from the Reference Model. Rich then discussed the work performed by each of the Focus groups. The External Interfaces group worked on requirements for operator-machine interface (OMI) and the minimum requirements for inclusion in the OS. A model for placement of the OMI services was also developed. The Multi-Language group considered requirements for the interface of both the Ada runtime environment and of other languages to the OS, developed Ada interface requirements and discussed the interface needs for Ada debugging and performance monitoring. An outline for discussion of the various issues was developed. This Focus group also considered extensibility of the OS, and the means by which applications could add new services to the system. It was suggested that such issues could be placed in a section discussing "non-standard" extensions. Input on these issues is sought from the working group at large.

The Fault Tolerance/Performance Monitoring group considered a number of requirements in their area. They developed a number of assumptions about the interfaces to a fault tolerant kernel, and

also concluded that the OS could probably not protect against defects in applications software. A triangle showing influences on the OS illustrated the point that security, fault tolerance, and performance considerations each pushed OS requirements in different directions; the challenge to the working group was to select a proper mix of the three. The Security focus group documented a large number of requirements, updating and enlarging the list from the previous meeting. The Real Time/Distributed Processing group identified over 400 requirements, and refined that list down to 112 that were of concern for NGCR. Near term plans for the Requirements Subgroup are writing an "initial preliminary first draft" of the Requirements document and the OCD, and to continue to develop and refine the OS requirements. Rich emphasized the need to get comments to Dan Juttelstad at NUSC.

Closing Comments

Following the Subgroup reports, CDR Barbour presented his wrap-up of the meetings accomplishments. Stating that progress is "phenomenal" for three meetings, CDR Barbour indicated his feeling that the OSSWG achieved a consensus on their goals on Tuesday, and that the Subgroups were making good progress on meeting their respective milestones. He is actively seeking offers to host meetings in 1990; one offer for a meeting in Honolulu is under consideration. The basic requirements for hosting are adequate rooms: a conference room or auditorium suitable for 70-90 people, plus additional rooms for Subgroup and Focus group meetings, and no cost to the Navy. Hotels are suitable, but a commitment from the hotel for meeting space may be difficult to get unless attendance can be guaranteed. Since "POSIX keeps popping up," CDR Barbour is also looking for volunteers to monitor the progress of POSIX, which holds 4 meetings per year. He hopes to brief the POSIX working group on NGCR requirements at their fall or winter meeting.

Stephanie Alba from Booz-Allen spoke briefly about coordination for the next meeting, which will be 1-3 August at the Naval Surface Weapons Center, White Oak, MD. Due to security issues at that facility, preregistration at least a week in advance is needed. Attendees who are foreign nationals must preregister as soon as possible.

Tricia Oberndorff wrapped up the meeting with several topics. She showed the schedule of meetings for 1990, which have been planned to avoid other meetings of interest to OSSWG members, and would like to be notified of any other conflicts with the schedule. Currently one briefing is scheduled for the August meeting; the topic is the Army Secure Operating System (ASOS), a project which addressed requirements similar to those of NGCR. Various working group members have requested tutorials on such topics as Ada, security, fault tolerance, and Open Systems Interconnection (OSI). Tricia may also schedule one or more such tutorials for August. Discussion occurred on how to make those

available to the most members of the group; suggestions included holding tutorials in the evening and/or video taping them. Tricia emphasized the need to get comments to the appropriate individuals on the various topics under consideration by the OSSWG, and strongly encouraged members to get email accounts and use them. She also reminded group members that they must focus on OS interfaces and not implementation. In closing, she congratulated the group on their program and thanked them for their efforts.

NEXT GENERATION COMPUTER RESOURCES PROGRAM

OPERATING SYSTEMS STANDARDS WORKING GROUP

MEETING 1-3 AUGUST 1989

MEETING MINUTES

The fourth meeting of Next Generation Computer Resources (NGCR) Program Operating Systems Standards Working Group (OSSWG) was held August 1-3 at the Naval Surface Warfare Center (NSWC), White Oak. Approximately 50 representatives of Government, industry and academia gathered to hear presentations given to the plenary group Tuesday morning. The remainder of Tuesday afternoon, Wednesday, and Thursday were allocated to Subgroup working meetings with wrap-up presentations given late Thursday morning.

Tuesday, 1 August.

CDR Rick Barbour provided welcoming remarks reminding the group of the upcoming dates for OSSWG deliverables. He then showed a chart of the OSSWG organization divided into Subgroups, illustrating the respective focus groups within each. The floor was then turned over to the Subgroup chairmen for their status reports.

Rich Bergman of the Requirements Subgroup stated that (at this gathering) they would be reviewing and refining the Draft Requirements Document and cross-checking requirements to the Abstract Model (a.k.a. Reference Model). Phil Hwang, Available Technology Subgroup chair, planned to complete Draft Version 0.2 of the Technology Report and continue work on the OS Survey and OSSWG definitions document. Tom Conrad, Approach Subgroup chair, announced the completion of Version 0.2 of the Evaluation Process and Version 1.01 of the Reference Model as well as continued work on the definition of the Evaluation Process.

The first invited speaker to address the plenary session was John Preusse, U.S. Army CECOM, who talked about the Army Secure Operating System (ASOS). Mr. Preusse outlined the classified information difficulties currently being experienced while processing ATCCS wide area communications. He discussed how overlooking security issues during the ATCCS operational concept has resulted in cumbersome data communications, requiring manual reading and retyping of information because of a lack of a secure computer interconnection system wide. Two possible alternatives to this situation were reviewed, multilevel security implemented in "trusted guards" between distinct system components, and incorporating a COTS multilevel secure operating system to integrate the individual systems in the command and control network. Both were determined as having unacceptable disadvantages. Mr. Preusse then presented ASOS as the CECOM-developed OS for multilevel security and real-time response. ASOS was designed for A1 security, supports Ada applications, and implements real-time OS features. An additional ASOS feature is "program sealing" for virus/worm protection. Mr. Preusse outlined the current schedule for porting ASOS to the ATCCS system and then touched upon some of the issues current to this decision. These included the performance issues due to multilevel security and the functional interface differences between ASOS and Unix application program environments. The major advantages of implementing ASOS were included as Mr. Preusse's

concluding remarks: ASOS meets the A1 multilevel security requirements for ATCCS while providing real-time features and protection against malicious software; even though there are functional differences, ASOS does interface with Unix-based software in addition to Ada applications programs; and there will not be an ASOS-equivalent COTS OS because industry lacks the incentive to develop one beyond the B2 level. To end the presentation, Mr. Preusse addressed the group with several of the "lessons learned" with ASOS development. He stressed that security must be designed into a system at initial concept. He also emphasized the importance of the early involvement of security evaluators (i.e., National Computer Security Center) in order to more easily certify the implementation of security requirements.

The next presenter was Dr. Cy Ardoin, who gave an overview of the issues confronting the Real-Time Embedded Systems and Distributed/Parallel Systems Working Groups of the June Ada 9X Workshop. The Real-Time group is concerned mainly with the speed of the operational code, the size of the memory required for operational code, and the correctness of the program. Dr. Ardoin then quickly touched upon the specific issues being addressed by the Real-Time working group, the Ada requirement involved, and possible work-arounds or solutions to the problems. The problems included: reliably turning off run-time checks, control over parameter transmission, task type interrupts, bit operations, volatile memory sharing, fixed point accuracy, and unsigned types without overflow. The Distributed/Parallel Systems working group focused on Ada in parallel and distributed processing architectures, specifically, language support for future Ada projects implemented on these types of systems. For this portion of the brief, Dr. Ardoin identified problem areas addressed by the working group and potential new Ada language requirements that address the problem. The problem areas included: types of distributed or parallel architectures, program partitioning, support for fault tolerance and dynamic configurability, inter-task communication, adaptive scheduling, memory management, time representation in distributed systems, identification of raised exceptions, and identification of control threads. To conclude his presentation, Dr. Ardoin outlined the Ada 9X Project requirements development process and identified the following sources of information regarding 9X progress: the Ada 9X Project Office, the Ada Electronic Bulletin Board, the Ada 9X Project Plan, and the Ada 9X Project Requirements Workshop.

Following the Ada 9X presentation, Tricia Oberndorf led the OSSWG in a technical discussion attempting to better refine the domain of the OS Standards within the NGCR scope of future Navy platforms. The first topic opened for discussion was the distinction between a Network OS vs. a Distributed OS. The group proceeded to engage in several rounds of lively discussion while trying to establish a definition for both concepts. Although they failed to reach a consensus on exact definitions, the OSSWG did

agree that the OS should allow for global resource sharing on a platform basis. This prompted questions regarding the types of OS functions that would be available at each "node" of the system. Two differing arguments arose. The first was that, since the functions needed by the individual platform systems were far ranging and often dissimilar, the system architecture should be the driving design force and the OS should be tailored to that. The second argument was that there needed to be a defined set of functions provided at each node, whereas the system designers would be required to use these functions when determining the system architecture. (Editorial note: this is much like the "who came first" chicken vs. egg debate). This discussion spawned a debate between the OS Standard defining a "loosely coupled" or "tightly coupled" system. It was noted that, in some instances, time criticality and performance will demand a tightly coupled definition of OS services. The final OSSWG consensus was to consider the Ada 9X method of stating requirements that allow for implementation specific approaches and have the requirement wording appear as "the system shall not preclude... ."

Tricia also informed the group of the most recent NGCR co-chair meeting with the other working group co-chairmen. She emphasized that the NGCR Program Office still needed to establish an NGCR interface definition among the individual components (OS, SAFENET, Backplane Bus). Tricia told the OSSWG that she had shown the other NGCR working group chairmen the graphic representation of OS system interfaces to their components and encouraged them to do the same for their particular efforts. Multiple views (from the working group chairmen) are needed for a consensus on a single, harmonious interface definition between components.

The OSSWG plenary session then ended, with Subgroup meetings scheduled for the remainder of the afternoon.

Wednesday, 2 August

Subgroups reconvened for all day sessions.

Thursday, 3 August

The Subgroups reconvened Thursday morning from 8:00 am to 10:30 am. After the Subgroup meetings adjourned, the working group as a whole reconvened to hear Subgroup reports and wrap ups from CDR Barbour and Tricia Oberndorf.

Requirements Subgroup

Rich Bergman gave a briefing on the progress of the Requirements Subgroup. He stated that their progress had become somewhat impeded by unresolved issues, the most significant being the still-undefined relationship of interface requirements to hardware, architecture, and application software requirements.

Additional issues are the definition of the Reference Model hierarchy and the questions concerning the relationship between the OS, Backplane Bus, and SAFENET standards. The final set of remaining issues addressed the topics discussed by the Tuesday plenary session; resource management and definition of a minimum/maximum set of OS services. Rich then identified the current schedule of events for the Requirements Subgroup. The Requirements Document reissue and the Draft OCD are due to be completed September 8.

The Available Technology Subgroup chairman, Phil Hwang, noted their accomplishments. Version 0.2 of the Technology Report had been completed and the OS Survey had been updated. Draft 0.3 of the Technology Report is scheduled for August 18 with the last Draft due September 14 and the Final slated for September 30. Phil also called attention to the 1989 Workshop on Operating Systems For Mission Critical Computing, which will be held September 19-20 at the Marriot in Greenbelt, Maryland. Phil, Tricia Oberndorf, and CDR Barbour will serve as committee Co-Chairs for that workshop.

Tom Conrad then presented the status of the Approach Subgroup. They had delivered the revised Reference Model (Version 1.01) to the OSSWG, developed a model of the Evaluation Process, produced a revised draft (0.2) of the Evaluation Process Report, outlined the content of the OSSWG semiannual report, and established a new focus group to study the issues of NGCR Conformance Testing for the OS Standards. Tom then presented graphical views of the Evaluation Process from several levels of detail. The detailed view of the Evaluation Process showed the abstraction of OS evaluation criteria into technical service classes (technical OS issues), programmatic issues, and representative application domains (families of services, etc.). The selection of an OS Standard will be derived from the filtering algorithm's final result, determined from the scores awarded in these three areas. The specifics of this process are included in Tom's presentation, included as an attachment to these minutes. Tom concluded his briefing with a list of actions which need to be completed before the Evaluation Process can become finalized.

Wrap-up presentations were then given by CDR Rick Barbour and Tricia Oberndorf. Rick told the group that he was impressed with the current progress of the Subgroups. Tricia took this opportunity to gain OSSWG consensus on the direction the OS interface Standard is heading. She noted that we are not "mixing and matching" functional modules to make up a working OS (i.e. Vendor A's scheduler with Vendor B's interrupt handler), but, instead, we must define the interfaces external to OS functions and restrict our focus to related issues. Tricia suggested that NGCR OS vendors should provide a minimal set of interface services, selectable at SYSGEN. This triggered arguments to what constitutes a "minimal set". She also reiterated that the OS interface concept is still missing a hierarchial picture demonstrating the management of system resources, such as is

illustrated by the SRAX in the Reference Model. On the subject of Conformance Testing, Tricia pointed out that conformance testing is not performance testing, but that a set of performance benchmarks should be made available from the NGCR Program Office. On the same subject, it was noted that, because the same amount of services may not be required from every NGCR OS Standard Product, levels of Conformance Testing may be required similar to the way layers of OSI Protocols are certified. CDR Barbour then called the meeting to an end, advising the group that the next OSSWG will take place September 12-14 again at the NSWC, White Oak, MD. He also changed the date of the December OSSWG, being held in San Diego, CA, to 12/12 - 12/14.

NEXT GENERATION COMPUTER RESOURCES PROGRAM

OPERATING SYSTEMS STANDARDS WORKING GROUP

MEETING 12-14 SEPTEMBER 1989

MEETING MINUTES

The fifth meeting of Next Generation Computer Resources (NGCR) Program Operating Systems Standards Working Group (OSSWG) was held September 12-14 at the Naval Surface Warfare Center (NSWC), White Oak. Approximately 65 representatives of Government, industry and academia gathered to hear presentations given to the plenary group Tuesday morning. The remainder of Tuesday afternoon, Wednesday, and Thursday were allocated to Subgroup working meetings with wrap-up presentations given late Thursday morning.

Tuesday, 12 September.

CDR Rick Barbour provided welcoming remarks and asked OSSWG members to check the current mailing list for accuracy regarding their name, address, phone/fax, and ARPANET location. He then set the tone for the meeting with a slide encapsulating the OSSWG charter. He reminded the group that they were there to adopt or establish a commercially based family of operating systems interface standards. This should be accomplished by determining Navy needs in the operating systems area, examining existing or proposed commercial operating systems, selecting a "best fit" based on Navy needs and operating system suitability, and recommending appropriate changes to this "best fit" for Navy-wide incorporation. CDR Barbour then showed the group the current milestone schedule and emphasized that time was growing short to the due date of the OSSWG Evaluation Report/Recommendation to the NGCR Program Office. Noting that it was time to "stop cutting bait and start fishing", CDR Barbour encouraged each of the Subgroups to meet their individual milestone commitments and coordinate their efforts to produce an effective evaluation package for a list of candidate OS interfaces in time for the January 1990 meeting. The floor was then turned over to the Subgroup chairmen for their status reports.

Phil Hwang, Available Technology Subgroup Chairman, gave a quick summary of that group's current status and outlined plans for this meeting. He noted that very few comments were generated concerning the latest version of the Technology Report, but those that were generated were excellent and incorporated into the new version, which was available for review. The most significant comment was that the report itself was becoming too large, almost unmanageable. Phil also related that the OSSWG Glossary had been updated. He then reminded the OSSWG of the upcoming (September 19-21) MCOS workshop (Co-sponsored by SPAWAR) taking place at the University of Maryland and he encouraged members of the group to attend. For this meeting, Phil planned to "split-up" the Technology Report into a number of documents and isolate the Abstract Model survey of OSs for use in the screening process. Phil also announced plans to incorporate the proceedings of the MCOS workshop into the Technology Report. Other plans for the group included finalizing the different sections of the Technology Report and delineating the initial screening process that will be used as a "first cut" for determining candidate OSs. Phil reported that it will be no problem for the Available Technology

Subgroup to adhere to its milestone schedule.

Steve Howell then spoke for the Approach Subgroup. He recapped the product responsibilities of that group, highlighting finished products such as the OSSWG POA&M (March '89), OCD DID (June '89) and Reference Model (July '89). Comments are still being solicited for the Reference Model. Steve also illustrated the upcoming products of the Approach Subgroup, the Evaluation Process Report (due (draft) September '89, (final) January 90) and the Semi-Annual OSSWG Public Report (October '89, April '90). Since the last meeting, the Approach Subgroup has received and incorporated comments to the Reference Model. They have also released a new draft version of the Evaluation Process that reflects comments concerning the previous draft. Additionally, they have begun to assemble the Semi-Annual Public Report. For this meeting, Steve said that the Approach Subgroup would finalize the Reference Model, achieve a consensus on mapping requirements (generated by the Requirements Subgroup) to the Model, brief the current (draft) Evaluation Process to the OSSWG in hopes of resolving the remaining Evaluation Process issues, and assemble information toward the first version of the Semi-Annual Public Report.

The Requirements Subgroup Chairman, Rich Bergman, was not present at this meeting due to illness. Tricia Oberndorf gave a brief overview of that groups status and plans. Tricia stated that the group would not break into smaller groups this session because of the need to refocus the Requirements Document. She said that the document required much work in order to get it ready for the October delivery date and the primary emphasis would be to orient the requirements to those representative of an OS interface versus those of an OS implementation.

Emily Siarkiewicz (USAF RADC) and Rammohan Varadarajan (Odyssey Research Associates) briefed the group on the Air Force's Secure Distributed Operating System (SDOS), which is still under development. A goal of SDOS is that autonomous hosts cooperating at an applications level can share common resources at local and remote sites in a secure C^2 environment. SDOS is modeled after BBN's Cronus. Its environment is a layered, two-tiered architecture where it performs object and process replication in a client/server fashion. The typical SDOS distributed system combines a "Client" and a "Manager" with each kernel (or resource node). A node Manager creates and maintains an object database that outlines the specific operations available on that node of the distributed system. They also enforce MAC and DAC and perform Audit functions. Services for remote resources are performed by Clients. Clients invoke operations on objects (desired resources in another node's Manager data base) through the use of a library consisting of low level communications primitives, an intermediate level program support library, and high level library calls. Mr. Varadarajan suggested that the SDOS Security Policy, consisting of a Mandatory Policy, a Discretionary Policy, and a Configuration Policy, would be applicable to the NGCR OS interface standard. A working SDOS system is expected by September, 1990

The next presentation was given by the Approach Subgroup. Tom Conrad, Approach Subgroup Chairman, emphasized that there will only be two more meetings between this one and the the one where the OSSWG is scheduled to evaluate candidate OS Standards. He stated that more input to the Evaluation Process is needed from the other OSSWG Subgroups. Tom announced his plans to go over the Evaluation Process as it currently stands, identify significant issues to be resolved, and hold an "open forum" discussion immediately after the presentation. Tom then introduced Steve Howell to review the current Evaluation Process Model. Steve showed an overview in terms of input, output, and key attributes. Inputs consisted of the evaluators themselves, the candidate OS interface specifications, the criteria that will be used to evaluate the candidates, and the weights assigned to criteria according to their relationship to key attributes. The output was defined as a series of scores rating each candidate interface specification against each key attribute. The key attributes were identified as Service Classes, Programmatic Issues, and Representative Application Sets. Steve then illustrated the mapping of the components of the Evaluation Process Model. He explained that hundreds of low level requirements, identified by the Requirements Subgroup, would be translated into quantifiable criteria, weighted according to their relevance to service classes, programmatic issues, or representative applications. The final score for a candidate specification will be derived by evaluating the specification against the criteria established for each of these attribute areas. After a series of questions, Tom Conrad reviewed the remaining open issues of the Evaluation Process. These included:

- Agreement on the overall process -
  -- Does the OSSWG concur?

- Agreement on representative application set -
  -- There are many different types on a variety of Navy platforms. Which ones do we incorporate? How do we represent them?

- Process to define evaluation criteria -
  -- Hundreds of requirements have to be translated into criteria mapped to service classes and programmatic issues. How many service classes?

- Process to define raw weights -
  -- What is process to define them? What are the metrics?

- Process to define raw scores -
  -- What do the numbers mean and how should they be interpreted? How do we put them in perspective?

- Filtering process -
  -- Do we keep the highest scoring candidate for a single representative application? How do we minimize inherent biases or misunderstandings of the criteria?

Evaluator issues -

-- What should be the qualifications of the evaluators? Do all evaluators score all areas of the candidate specifications? Should there be a minimum number of evaluators for each criterion? Do we employ the use of experts?

- Form of results from process -

-- Do we want to give a one number result? Should the results be graphed as a curve?

Tricia Oberndorf then continued the open forum discussion and reinforced to the group the fact that these issues must be resolved quickly. She also stressed that the OSSWG efforts will result in an interface baseline, and, in all likelihood, the candidate OS specifications will not meet all the requirements of this baseline. A major reason for this is that we are looking for an operating system interface specification and that the majority of candidates are implementations of operating systems, not just their interfaces. The capabilities of the family of interfaces (the "library") to be incorporated as the NGCR standard will most probably far exceed the capabilities of any single candidate OS. However, Tricia then strongly stated that we are not out to design or develop a standard, our purpose is to adopt one and modify it as little as possible.

After receiving encouragement from CDR Barbour to resolve the Evaluation Process issues during this meeting of the OSSWG, the group broke for lunch and reconvened into Subgroups.

Wednesday, 13 September

Subgroups reconvened for all day sessions.

Thursday, 14 September

The Subgroups reconvened Thursday morning from 8:00 am to 10:30 am. After the Subgroup meetings adjourned, the working group as a whole reconvened to hear Subgroup reports and wrap-ups from CDR Barbour and Tricia Oberndorf.

Requirements Subgroup

Dan Juttelstad presented the progress of the Requirements Subgroup. He reported that they had modified the Requirements Document to conform with the Reference Model and also to reflect interface requirements rather than OS implementations. Due to the emphasis placed on reorienting the Requirements Document, the OCD was not updated. The revised Requirements Document reflecting the new format and emphasis will be available 9 October. Dan said that the group will then fold the Reference Model and the new Requirements Document into the appropriate sections of the OCD. In closing, Dan told the group that the Requirements Subgroup will

meet for 5 days (16-20 October) at the next OSSWG (in Newport, RI) instead of the usual 3 days.

Available Technology Subgroup

Phil Hwang reviewed the accomplishments of the Available Technology Subgroup. Most notably, they restructured the Technology Report into several documents of a more manageable size. They also began to define the screening process that will be used to select OS standard candidates for evaluation. He expressed concerns over linking the screening process to the actual Evaluation Process. Phil also showed the group the results of a "first run" screening, and then presented a more definitive list of the 8 characteristics that the group will be using in the final screening process.

Approach Subgroup

Tom Conrad presented the efforts of the Approach Subgroup. He reported that they had made some progress in coordinating the Service Class definitions with the Requirements Subgroup. They also established a detailed schedule for implementing the Evaluation Process (that schedule is included in this package). Other progress included the determination of a weighting process and a filtering algorithm (for the raw scores of the functional and programmatic evaluations). Tom then tackled the issue of how to best prepare the evaluators for the January meeting where the evaluation will take place. He suggested that each evaluator receive a complete evaluation package (candidate specifications, evaluation criteria, scoring sheets, etc.) several weeks before the meeting in order to start the process before arriving. The meeting could then be used for discussions or to ask questions of the various representatives of the candidate specifications. The evaluators could then go home and complete the evaluation in the next several weeks, finishing by 16 February '90. The draft selection report would be completed in mid March, with the final delivered to the NGCR Program Office in the beginning of April. Tom then expanded on the filtering algorithm. He said that due to the shear number of evaluators and number of criteria resulted in the selection of a "mean score" as the filtering algorithm. Tom then described the results of the "weight" issues concerning the Evaluation Process Model. One set of weights identifies the relative importance of evaluation criteria to a particular service class. This weight set will be derived by the OSSWG during the December meeting where each criterion's contribution to a service class will be scored on a 0-10 scale. These raw scores will then be averaged and normalized to produce weight set "1". Tom noted that it was extremely important to accurately map the OS requirements (derived by the Requirements Subgroup) to the evaluation criteria. The second set of weights identifies the relative importance of OS service classes to a representative application domain. These were derived by the Approach Subgroup at this meeting. The criteria for establishing evaluators were

also decided. To be eligible, you must be technically oriented (versus marketing) and have participated in at least two OSSWG meetings by the end of the December OSSWG (limit: 2 evaluators per company/organization). It was also determined that all evaluators must evaluate all the candidates, but not against all the criteria. Tom then emphasized the enormous task each of the evaluators will be facing. He presented figures representing the potential number of pages of documentation (5000) to be read and the number of score sheets to be filled out (340) by each evaluator. Tom also revealed that the Approach Subgroup had revised their approach to identifying representative applications. The new approach consists of 8 application domains, chosen to exhibit differing service class requirements. More information on these will be made available at a later date. Tom concluded his presentation with a list of "Things to Do" and "Things We Need". These lists are included as part of this package.

The final wrap-up comments were given by CDR Barbour and Tricia Oberndorf. Both congratulated the efforts of the OSSWG at this session and again stated that our purpose is to select an industry standard that meets the Navy's needs for Operating Systems interfaces. Tricia also encouraged the OSSWG to make comments to the documents that are sent out on the net and to take the commitments made to the OSSWG seriously.

The next OSSWG will meet 17-19 October (16-20 October for the Requirements Subgroup) at NUSC in Newport RI. Everyone is reminded to send a Visitor Request to the NUSC security office.

NEXT GENERATION COMPUTER RESOURCES PROGRAM

OPERATING SYSTEMS STANDARDS WORKING GROUP

MEETING 17-19 OCTOBER 1989

MEETING MINUTES

The sixth meeting of Next Generation Computer Resources (NGCR) Program Operating Systems Standards Working Group (OSSWG) was held October 17-19 at the Naval Underwater Systems Center (NUSC), Newport, Rhode Island. Approximately 45 representatives of Government, industry and academia gathered to hear presentations given to the plenary group Tuesday morning. The remainder of Tuesday afternoon, Wednesday, and Thursday were allocated to Subgroup working meetings with wrap-up presentations given late Thursday morning.

Tuesday, 17 October.

CDR Rick Barbour provided opening remarks and reminded the group about the upcoming December (12-14 in San Diego, CA) and January (22-26 in Mobile, AL) meetings. CDR Barbour then extended his congratulations to the recipients of the NGCR Backplane Bus contract award, Litton Corp., Raytheon Co., and Cable and Computer Technology (CCT) Inc. The presentation then focused on the meeting objectives. The group was instructed that it is time to begin preparations for the January, 1990 evaluation process. CDR Barbour stressed the importance of each participant understanding the scope of the remaining work and the short schedule. He then reviewed the individual subgroup products and what he expected the subgroups to accomplish by the end of this meeting. Specifically, the Requirements Subgroup needed to complete reviewing the requirements and finalize them for a new draft of the Requirements Document, the Available Technology Subgroup was instructed to produce a "first cut" Candidate list, and the Approach Subgroup was to further refine the evaluation process.

CDR Barbour then introduced the meeting's host, CDR John Reed, the NUSC Executive Officer. CDR Reed gave the audience a brief description of the NUSC mission, leadership abilities, and goals. He then discussed the NUSC locations in New London, Newport, and Andros island in the Bahamas in addition to providing an overview of NUSC expertise.

Subgroup Reports

Rich Bergman, Requirements Subgroup Chairman, announced that they had completed a first cut at Section 1 of the OCD. He also stated that they had further refined the Requirements Document and reviewed issues with the Reference Model, most notably, better resolution of SRAX and IRAX hierarchy. Rich then stated plans to bring the OCD closer to a first draft, incorporate the current list of requirements into the OCD, and justify each of the requirements for use as evaluation criteria.

Approach Subgroup Chairman, Tom Conrad, noted that his Subgroup had generated version 0.4.13 of the Evaluation Process Document, refined the Reference Model, and reworked the Representative Application Domain Set of the Evaluation Process

Model. Tom's intentions for this meeting were to have the Approach Subgroup finalize the scoring procedures, define the Representative Application Domains and their associated weights (weight set 2), and develop evaluation forms.

Jim Oblinger made the presentation for the Available Technology Subgroup, as Chairman Phil Hwang was not able to attend the meeting. Jim reported that the OS workshop held at the University of Maryland was very successful and the preliminary proceedings were currently available. He also said that the Technology Report had been restructured at the last meeting and was still being updated. The plans for the Subgroup at this meeting were to continue working on the Technology Report and complete the OS interface candidate prescreening.

The OSSWG was then given a presentation by Walter Shore, Motorola Corp., on the Open Real-time Kernel Interface Definition (ORKID). ORKID was developed by the VME International Trade Association (VITA). It is promoted by that group as an open real-time software interface that is not particular to any bus or hardware architecture. The objective of the ORKID standard is to provide a state-of-the-art open real-time kernel interface definition that allows users to create robust and portable code while allowing implementors the freedom to proliferate their compliant product. Walter explained an ORKID system as a collection of one or more interconnected nodes, with each node being serviced by a computer with an ORKID compliant kernel on which application programs run. Walter noted that a node is a single entity in ORKID, although it may be implemented as a multi-processor computer. Walter then reviewed the various features of ORKID, including naming and object identification, tasks and their operations, memory regions (for dynamic allocation) and partitions (for fixed allocation), semaphores, queues, events, exceptions, clocks, and timers. Walter finished the briefing with a presentation of current ORKID status. ORKID draft 1.0 has been available for public comment since July, 1989. Comments are reviewed at each ORKID subcommittee meeting, held approximately every 3 months. A final version of ORKID will be submitted for VITA approval within the next six months. Upon approval from VITA, ORKID will be presented to other standards organizations such as the IEEE and IEC.

The final briefing of the day was given by Tricia Oberndorf. The topic of Tricia's presentation was the current list of cross-working group (Backplane Bus, SAFENET, OSSWG, Conformance Test) issues. They are:

- The NGCR Model - There currently exists no overall NGCR model. It is not clear where and how the individual NGCR pieces fit together nor what such a model should look like.

- Real Time - The scheduling approaches of each NGCR standard need to be defined. The concept of a global clock must be explored to ensure consistency between the Backplane, SAFENET, and OS system components.

- External Interface and Boundary Definitions - Determine where the individual components (SAFENET, OS, Backplane Bus) begin and end. Each standard should be capable of standing alone as well as working together. NGCR standards should have the capability of operating in non-NGCR systems (e.g., OS working with a LAN other than SAFENET).

- System Distribution - How to accomplish load leveling and control management.

- Performance Monitoring - Will there be minimal NGCR configurations for the purpose of Conformance Testing? Will there be performance testing as well (much like the ACVC (conformance) vs. ACEC (performance) for Ada compilers)?

- Security - What are the formal security requirements for each standard? The NGCR Program should have a security model.

- Fault Tolerance/Recovery - How do we "warm start" to system configuration?

- Ada - The OS Interface Standard has Ada interfaces but SAFENET and the Backplane Bus Standards do not.

Tricia also touched upon other issues including: reconfiguration, real-time (non-intrusive) tests, message passing across the backplane, hardware and software prototyping, and nuclear survivability.

Wednesday, 18 October

Subgroups reconvened for all day sessions.

Thursday, 19 October

The Subgroups reconvened Thursday morning from 8:00 am to 10:30 am. After the Subgroup meetings adjourned, the working group as a whole reconvened to hear Subgroup reports and wrap-ups from CDR Barbour and Tricia Oberndorf.

Requirements Subgroup

Rich Bergman presented the list of accomplishments and future plans for the Requirements Subgroup. He stated that all the requirements in the Requirements Document will have been frozen by the end of the week. The December issue of the Requirements document will represent the final list. Additionally the requirements will reflect those of an Interface Standard vice an Operating System implementation. Rich also reported progress on the evaluation criteria and planned to have the criterion for each requirement defined before the December OSSWG meeting. In closing, Rich acknowledged that the SRAX and IRAX hierarchy issues

still need to be resolved.

Available Technology Subgroup

Jim Oblinger told the OSSWG that the major emphasis of his Subgroup's efforts for this meeting were spent on completing a prescreening of candidate OS interfaces. He then identified his view of Available Technology Subgroup activities for the next two OSSWG meetings. (Tom Conrad, Approach Subgroup Chairman, had independently assembled a list of remaining activities for each of the Subgroups, with slightly different associated dates and assignments.) Jim then recounted the specific ground rules that were employed in the prescreening process. Each of the candidates was evaluated by the Available Technology Subgroup against both positive and negative pre-selection criteria. After evaluating each candidate, an overall rating was assigned reflecting the most prominent positive or negative criteria associated with that particular candidate. The final list of pre-screening candidates consisted of those with an overall positive rating. This first cut candidate list consists of (alphabetically): Alpha, ARTX, CRONUS, iRMX, MACH, ORKID, POSIX, TRON, Trusted MACH, and VDIST. Jim cautioned the group that this list is still subject to change, with the final list to be determined by OSSWG consensus at the December meeting.

Approach Subgroup

Tom Conrad began his presentation with a slide showing that the Approach Subgroup had finalized the scoring process as well as completed the descriptions of Representative Application Domains and their associated weights (weight set 2). He then showed the group the results of an initial poll that illustrated the number of OSSWG attendees that had voluntarily chosen service classes by which they would evaluate candidate OS interfaces. The results indicated that several of the service classes were not well represented by potential evaluators. Tom noted this and then informed the OSSWG that service class representation would most likely be a combination of elected and assigned responsibilities to the evaluators. Tom then went over a detailed milestone schedule depicting the activities required to establish evaluator's packages, establish weight set 1, prepare for the January meeting, and produce an evaluation/recommendation. The focus of Tom's discussion centered on the fact that most of the activities in each schedule were on the critical path with little or no room for slippage. Tom finished his presentation with an examination of each of the 8 Representative Application Domain descriptions and the relative weighting of each against the current 15 service classes

Wrap-up comments were given by CDR Barbour who congratulated the efforts of the OSSWG at this session and again stated that, although significant progress has been made, there is much work to be done.

The next OSSWG will meet 12-14 December at the Vacation Inn in San Diego, CA. Everyone is reminded to make room reservations no later that November 12.

NEXT GENERATION COMPUTER RESOURCES PROGRAM

OPERATING SYSTEMS STANDARDS WORKING GROUP

MEETING 12-14 DECEMBER 1989

MEETING MINUTES

The seventh meeting of Next Generation Computer Resources (NGCR) Program Operating Systems Standards Working Group (OSSWG) was held December 12-14 at the Vacation Inn, San Diego, California. Approximately 70 representatives of government, industry and academia gathered to hear opening remarks Tuesday morning. The remainder of Tuesday was devoted to the development of Weight Set 1 for the individual service classes. Wednesday was allocated to Candidate presentations. Thursday's agenda consisted of discussions on remaining issues, a dry run brief/evaluation of a non-candidate OS interface (CAIS-A), and the announcement of the final list of candidates for the NGCR OS Interface Standard.

Upon check-in for this conference, the registrants were asked to inspect the OSSWG attendance record to verify their eligibility to participate in the OSSWG evaluation process. Eligible evaluators were then asked to select the service classes that they would be responsible for scoring.

Tuesday, 12 December.

CDR Rick Barbour provided the opening remarks. The group was instructed that preparations for the January, 1990 evaluation process would be finalized at this gathering. CDR Barbour then gave the potential evaluators instructions to ensure that each had selected specific service classes to evaluate in conjunction with submitting a formal intent/commitment letter to Tricia Oberndorf. CDR Barbour stressed that it was absolutely necessary to satisfy these conditions in order to be considered eligible to perform candidate evaluations. He then discussed the remaining schedule and provided information on the next OSSWG meeting, scheduled to take place in Mobile, AL.

Rich Bergman, Requirements Subgroup Chairman, informed the group of the weight assignment schedule that would be followed for the remainder of Tuesday. He emphasized the shortness of the schedule and advised the group that the Tuesday session would not end until all of the Weight Set 1 assignments were made.

Dr. Carl Schmiedekamp then gave a short presentation on Weight Set 1, showing group assignments (the OSSWG was to be split into two groups, each responsible for assigning weights to six service classes), instructions for weighting criteria, and an explanation of how a weight would be determined for each criterion within a service class.

The remainder of the Tuesday morning session involved Tricia Oberndorf leading the plenary group in assigning weights to each of the evaluation criteria in the General Requirements service class. The format consisted of Tricia reading the definition of each requirement in that service class, the associated evaluation criteria and corresponding weighting guideline, and rationale, sometimes followed by group discussion. Each participant then assigned a weight (an integer value from 0 to 10) to that particular requirement, with the understanding that blank entries would receive a default weight of 5.

Due to time constraints and the large number of evaluation criteria, the plenary session was then divided into two groups in order to conduct weight assignments in a parallel manner for the Tuesday afternoon session. One group assigned weights to evaluation criteria in the following six service classes: File Interface; Generalized I/O Interface; Network and Communication; Reliability, Adaptability, and Maintainability; Resource Management Interface; and System Initialization and Reinitialization. The other group addressed the following six service classes: Event and Error Interface; Process Management Interface; Project Support Environment Interaction; Synchronization/Scheduling Interface; Time Services Interface; and Ada Language Support Interface. The Capability and Security service class criteria were assigned weights separately by a group of security experts.

Wednesday, 13 December.

The entire Wednesday agenda consisted of scheduled presentations by representatives of OS Interface Standard candidates. These were:

Alpha	-	Dr. Doug Jensen, Concurrent Computer Corporation
ARTX	-	Dr. Dado Vrsalovic, Ready Systems Corporation
CRONUS	-	Ken Schroder, BBN Systems and Technologies Corp.
iRMX	-	Van Kane, Intel Corporation
Mach	-	Rod Johnson, Open Software Foundation (OSF)
ORKID	-	Dick Vanderlin, Motorola
POSIX	-	Dr. Doug Locke, IBM Corporation
MTOS	-	Carol Sigda, Industrial Programming Inc. (IPI)
SDOS	-	Ken Schroder, BBN Systems and Technologies Corp.

Each presenter was allotted the same amount of time for a presentation followed by a question and answer period.

Thursday, 14 December.

The Thursday session was opened with general discussions, where all attendees were encouraged to bring forth any subject pertaining to the current selection process. CDR Barbour prompted the audience by stating that, unless there were objections from the audience, the final list of candidates would consist of those that were presented the previous day. When no one voiced an objection to the candidate list, CDR Barbour brought up the topic of performance requirements. He stated that the OSSWG was tracking many performance issues, but purposely avoiding in-depth discussions concerning them to allow a better focus on identifying

the interfaces and related requirements that will define the basis for the OS interface standard(s). CDR Barbour speculated that it is conceivable that NGCR-certified OS products may be subject to performance testing as well as conformance testing (much the same as Ada compilers are tested by ACVC and ACEC suites). He also touched upon the concept of an accompanying OS interface standard addressing performance criteria. For the benefit of some in the group who were not familiar with NGCR Program plans, CDR Barbour gave a brief description of the prototyping concept and how it applies to OSSWG efforts.

Tricia Oberndorf then assisted CDR Barbour in answering several questions, the most significant being "If the OS selection process yields multiple standards (vice 1), how will they eventually be merged into a single standard?" Tricia replied that it is a strong possibility that the OS interface selection will consist of several standards. She added that we have discussed the "family of interface sets" concept in past meetings, but, for now, the OSSWG consensus is to wait and see what comes out of the evaluation.

The majority of the afternoon session was dedicated to a "dry run" brief/evaluation presented by Gary Pritchett of Softech. The subject of the evaluation was CAIS-A, chosen, in part, because it is not a candidate for OSSWG selection. Mr. Pritchett instructed half of the attendees to evaluate CAIS-A against the criteria of one service class, with the other half evaluating it against a different service class. He then gave a presentation highlighting the features of CAIS-A in each of the service class areas, allowing for questions from the evaluators. The purpose of the exercise was to give OSSWG evaluators first-hand experience in using the tools of the upcoming evaluation (candidate documents, presentation materials, the OSSWG Requirements Document, scoring sheets, etc.) as well as to gauge how well a candidate can address all the required evaluation criteria in a relatively short period of time. (Mr. Pritchett gave an overview of CAIS-A and then focused on 2 service classes in roughly 3 hours)

The Thursday session concluded with a discussion of Subgroup business, followed by a presentation of the final candidate list and wrap-ups from CDR Barbour and Tricia Oberndorf.

Rich Bergman, Requirements Subgroup Chairman, announced that version 2.0 of the Requirements Document, including the new Security section, would be finished 12/20/89. This is the version that will be used in the evaluation and will be sent to all evaluators and candidates.

Dr. Karen Gordon, Available Technologies Subgroup, informed the OSSWG that Mach and TMach would be presented as a single candidate, as would CRONUS and SDOS. She also identified the reproduction and dissemination of proprietary documentation as a potential stumbling block that needs to be addressed prior to the evaluation. Dr. Gordon's final point was that candidates who do not provide a matrix cross-referencing their documentation to the

Requirements Document risk receiving low scores if the evaluators have a difficult time locating information. The final candidates for evaluation were announced as (alphabetically): Alpha, ARTX, CRONUS/SDOS, iRMX, Mach/TMach, MTOS, ORKID, and POSIX.

Tom Conrad, Approach Subgroup Chairman, focused his comments on the evaluation process. Tom noted that a total of 59 evaluators (17 Navy, 42 Industry) had received service class assignments for the evaluation. He also provided a slide depicting the number and type (Navy or Industry) of evaluators assigned to each service class. Tom then showed the group the list of Programmatic Issues that the OSSWG Navy members will use to further evaluate the candidates. These were: Public Domain Interfaces, Navy Influence, Maturity/Confidence, Documentation, Timeliness, User Influence, Economics/Cost, and Commercial Acceptance. Tom's slides are included with these minutes.

Tricia Oberndorf provided an explanation of how to evaluate the candidates according to the "Distributed Systems" requirements. She emphasized the need to evaluate the candidates according to how the OSSWG "SRAX" (from the Model) requirements are met. The difficulty of this is that "distribution" requirements are not isolated to a single service class, rather, distribution requirements fall across many service classes.

CDR Rick Barbour then provided his "wrap-up" comments. Appropriate authority allowing the reproduction and dissemination of candidate copyrighted/proprietary information for OSSWG use is being obtained and should not cause a problem at the January evaluation. Of note, the next NGCR OSSWG meeting will be held January 22-26 at the Stouffer Riverview Plaza Hotel, 64 Water Street, Mobile, AL 36602. He emphasized the importance of the meeting to the evaluators, as the primary agenda will consist of presentations from all the candidates. The evaluators will have an opportunity to ask specific questions to assist them in their evaluation. CDR Barbour then congratulated the group and encouraged continuation of the same successful effort.

NEXT GENERATION COMPUTER RESOURCES PROGRAM

OPERATING SYSTEMS STANDARDS WORKING GROUP

MEETING 22-26 JANUARY 1990

MEETING MINUTES

The eighth meeting of Next Generation Computer Resources (NGCR) Program Operating Systems Standards Working Group (OSSWG) was held January 22-26 at the Stouffer Riverview Plaza Hotel, Mobile, Alabama. Approximately 55 representatives of government, industry and academia gathered to hear opening remarks Monday morning. The Operating System Interface evaluation process was reviewed Monday morning. The remainder of the week was devoted to candidate presentations correlating the OSSWG requirements to the candidate operating system. A library of the candidate documentation was made available during the week and time was alloted on Wednesday morning for evaluation purposes.

Monday, 22 January.

CDR Rick Barbour provided the opening remarks. He first introduced Neil Henderson of host Litton Data Systems. Neil welcomed everyone and provided a brief orientation of Mobile. CDR Barbour then provided the meeting objectives to initiate the evaluation process, receive candidate briefings, and discuss the March meeting. The agenda for the week was presented and it was announced that a library containing copies of the candidate documentation would be made available all week. The next meeting was announced to be held at Naval Surface Weapons Center (NSWC), White Oak, MD.

Tricia Oberndorf announced that a discussion on security requirements for all those evaluating this service class will take place on Tuesday. Also, Tricia encouraged participation by OSSWG members in the SIGADA 9X efforts. It is desirable to have the OSSWG point of view represented at the 9X meetings.

Approach Subgroup Chairman, Tom Conrad, presented a brief evaluation process status. He noted that the interactive evaluation tool for submitting scores was running and that the evaluation sheets had been updated. The Evaluation Process Document is in version 0.8 and is available for comment. A dry run of the evaluation process was performed at the San Diego meeting and was fairly successful. Details of the status were deferred to his presentation later in the morning.

Jim Oblinger presented the status of the Available Technology Subgroup, noting that the San Diego briefers were the final candidates chosen for the evaluation. The candidate presentations for this meeting had been identified as technically oriented. Information on the candidate systems was being collected and coordinated by Booz, Allen. Jim then explained the reasons for the fluctuation in the candidate list between the San Diego meeting and Mobile. Initially, CRONUS had been constrained in their ability to support the OSSWG evaluation effort and did not plan on supporitng the evaluation. However, about two weeks prior to this meeting, after talks with CDR Barbour, they reevaluated their position and were participating. Industrial Programming (MTOS), on the other hand, determined that they could no longer

support the OSSWG and had to drop out. Finally, Jim stated that the Available Technology Report, version 0.7, was available in the OSSWG repository for comment and identified the subgroup's desire to finalize the document.

CDR Barbour briefly clarified Industrial Programming's view point as to why they withdrew MTOS as a candidate. Their primary concerns centered around copyright and proprietary issues. After internal discussions, they felt unable to meet all the needs of the OSSWG.

Tricia Oberndorf, filling in for Rich Bergman who was not available due to travel constraints, gave the Requirements Subgroup status. The requirements were finalized at the San Diego meeting and are reflected in the version 2.0 document. Current efforts of the group center around folding the Requirements Document into the Operational Concept Document (OCD). The OCD is due in May for review.

After the subgroup reports, Tom Conrad explained in detail the entire evaluation process and what is expected from it. The most important upcoming milestone in the process is the actual scoring which is being kicked-off this week. It was desired that all computations be completed by 23 February 1990, thus making the 9 February 1990 evaluation deadline important. Tom noted the tight schedule that was needed to get the final report and recommendations to the NGCR Program Office by early April.

The status of the documentation mailings was provided. As of the meeting date, all evaluators should have received a Requirements Document, one whole set of Evaluation Sheets, and ORKID. ALPHA and POSIX were in the mail and should be received shortly. ARTX and IRMX were currently being reproduced for mailing, and CRONUS and MACH had just been received in Mobile. Tom reiterated that copies would be made available at the hotel. He then reviewed the evaluator assignments and noted that Weight Set 2 was computed and publicly available.

There are a total of 71 evaluators: 25 Navy evaluators and 46 industry evaluators. Each service class is well covered by the group, but any evaluator may evaluate additional service classes if they desire. However, if an additional service class is evaluated, it must be done for all seven candidates.

Submission of scores during the current meeting was encouraged, noting that changes would be allowed after initial submittal. There are three methods of submitting scores. The first is by paper, however, the due date was 2 February 1990 if this option were to be exercised. The second method is by editing electronic templates of the evaluation sheets, and the third method is by using the interactive evaluation tool. Methods two and three were to be explained by Carl Schmiedekamp later in the morning.

Tom presented the results of the dry run evaluation in San Diego. Service classes 9 and 13 were evaluated for a sample candidate. The results provided scores from 0 to 10, an average score, a maximum score, a minimum score, and a standard deviation. The dry run was useful in helping to get the bugs out of the process. A method for analyzing the rationale and comments sections had not yet been developed.

Tom then went into a step by step review of the entire evaluation process to provide a background to new members and a refresher for others. The basic steps follow:

- 110 initial candidates selected
- List reduced to 10
- List further reduced to 7
- Evaluators were identified
- Requirements of an Operating System I/F developed
- Candidates to be scored by evaluators current)
- Scores will be combined using two weight sets
- Candidate scores will be charted showing relationships
- Recommendations will be made to SPAWAR 324

Tom explained the use of the two weight sets. Weight Set 1 was developed in San Diego and will not be available to the public until all scoring has been completed. It relates the relative importance of specific criteria within a particular service class. Weight Set 2 provides weights for eight representative application domains, named after gemstones. The second weights, which are public, map the importance of service classes to each application domain.

Tricia Oberndorf briefed the group on what was expected at the March meeting. The results will be charted in a number of ways for analysis. It is expected that the analysis will not be easy. Much of the meeting will be spent writing up the results of the evaluation and the recommendations. Attendees will probably be asked to write sections of the report at the meeting. As a result, not much time is expected to be available for subgroup meetings.

It is envisioned that there will be several alternatives for selecting a baseline. Some of the possibilities include:

- 1 candidate as a single baseline (desired)
- Some combination of a kernel and an OS
- Several candidates, each good in specific areas
- Profiles (e.g., set for Real Time, set for Transactions)

The goal for the evaluation would be to obtain a single baseline, but all possibilities must be considered. It is expected to be a lot of work and everyone's participation will be appreciated.

Carl Schmiedekamp then presented the use and understanding of

the evaluation forms, along with the various methods of submittal. As stated previously, there are three methods of submitting forms: paper, editing a downloaded set of forms, and interactive evaluation tool. If submitting paper, the completed forms should be mailed to:

Dr. Carl Schmiedekamp
Code 7033
Naval Air Development Center (NADC)
Warminster, PA 18974-5000

If submitting forms using the second method, the following steps should be followed:

- Download a set of forms from the OSSWG archives (Carl will also be e-mailing a set of forms to each evaluator).
- Make copies of the appropriate service classes for each candidate.
- Edit the forms using any editor. Be careful not to change the base form, especially the symbols (e.g., *) needed by the parsing tool to process the forms.
- E-mail the forms back to OSEVAL@NADC.ARPA .

The forms can be found in the OSSWG archives under General, in a file called "all ^H_forms.text" for a full set of forms, or individual forms by service class may be downloaded in files following the format "sc##.text" (## is the service class number).

Finally, the interactive tool may be used to submit forms. The tool will enable partially completed forms to be saved for later editing and will submit completed forms. A drawback to the tool is that it is slow if the user is not connected directly to the NADC machine. The interactive tool can be used by logging into the OSSWG archives (OSSWG@NADC.ARPA, password=NEXTGEN) or working from an individual account on NADC.ARPA and invoking the tool using the full path /USR1/OSSWG/EVALS/EVALUATE . To prevent crashing the system, there is a limit of 6 users of the tool at one time.

Carl then conducted a poll of the evaluators to get an estimate on the probable methods of submission by the group. Six evaluators chose submission by paper, 24 chose editing their own copies of the forms, and 7 stated they will be using the interactive tool.

Next, Carl reviewed the instructions for filling out the evaluation forms:

- All fields are required in the evaluation form heading
- Service Class Numbers: already provided

- Evaluation name: fill in your name
- Evaluation ID: 4 digit number assigned
- Password: 6 characters assigned
- Candidate ID: must be one of the seven - ALPHA, ARTX, CRONUS, IRMX, MACH, ORKID, POSIX
- Scores: range 0 to 10 (no default scores).

It was noted that all scores count the same, even though a confidence level is requested. The Requirements Document states how to score the candidates and what a particular score should mean. This should help normalize the scores across evaluators. Other methods were considered, but they did not really provide meaningful information.

Carl also stated that the analysis will also provide the error spread in the data through two methods. The first method is a calculated standard deviation. The second method uses the confidence level put on the scoring sheets. An "H" (High) means that the evaluator feels their score is close to the actual score. An "M" (Medium) means that the evaluator feels they may be off by about 2 points, and an "L" (Low) means that the evaluator has low confidence in the score and may be off by as much as 3 points. No entry in this field will default to "M". This level will allow a more exact analysis of which scores are more accurate due to evaluator certainty.

The Rationale/References section is provided for meaningful comments that are directly relevant to the specific criterion being evaluated and the score given. There is no limit to the length of information that can be put here or in the Comments. The Comments section of the evaluation form is provided for important information relevant to other criteria or items, but not directly relevant to the score. These two sections will be used in the Evaluation report. Carl requested that those who do provide comments, write them as if they are explaining their point to someone else. This will enable easier incorporation into the final report.

Finally, the General Comments section at the end of each service class provides a place for the evaluators to express their feelings about the candidate for the service class as a whole.

An issue was raised regarding the absence of an explicit class for distributed requirements. Tricia Oberndorf responded that this was a point identified at the San Diego meeting. Although requirements for a distributed system appear in the General section, this doesn't give a direct weighting to distribution. After much consideration, it was determined that everyone should remember distribution when evaluating all criteria. Evaluators should always keep distribution in the back of their minds during the evaluation and use the comments and rationale sections to record the difference between single systems and distributed systems.

Tricia also noted that there are two types of distribution:

transparent and explicit. Transparent distribution means that the interface doesn't see or care how the distribution is handled. Explicit distribution is directed or influenced by the interface (perhaps through an explicit call). It is important to take both types into account when performing the evaluation.

Jim Oblinger led off the candidate presentations by explaining what was requested from each of the candidates. The candidates were asked to present each service class, one at a time, and then accept questions on that particular service class after it was presented. They were also asked to state why their candidate OS should be the one that the OSSWG selects as a baseline. Finally, they should try to relate how the NGCR components (terminology) correspond to the candidate components. Presentations were to be scheduled for four hours.

Before each candidate presentation, the following reminder was provided:

WE ARE HERE TO EVALUATE INTERFACES!
WE WILL BASELINE AN INTERFACE.
IMPLEMENTATIONS ARE INTERESTING AS PROOF-OF-CONCEPT ONLY

The OS Interface Standard candidate presentations then began for the remainder of the week. The first presentation beginning on Monday, 22 January in the afternoon.

ORKID - Richard Vanderlin, Motorola

Tuesday, 23 January.

POSIX - Jim Isaack, Digital Equipment Corporation; Fritz Shultz, OSF; Jim Hall, NIST; Steve Carter, Bellcore; Mike Cossey, DOE Oakridge; Doug Locke, IBM Corporation; Steve Deller, VERDIX Corporation

IRMX - Tim Saponas, Intel Corporation

Wednesday, 24 January.

The morning session was set aside for evaluators to use the candidate materials made available at the hotel. A presentation was scheduled for the afternoon.

Alpha - Doug Jensen, Concurrent Computer Corporation

Thursday, 25 January.

ARTX - Dave Nelson-Gal, Ready Systems Corporation

MACH/TMACH/RMACH - Brian Boesch, DARPA; Richard Rashid and

Friday, 26 January.

CRONUS/SDOS - Jim Berets, BBN Systems and Technologies Corporation

After the final presentation, a short wrap-up was presented. Slides from the wrap-up follow the minutes.

Tom Conrad noted the missing evaluators at the meeting and identified methods to get copies of the presentation materials to these evaluators. Tom then reinforced some important considerations when performing the evaluation. He presented the Evaluator's Commandments slide. Noting the short time allowed for the evaluation by the schedule, it was announced that the 2 February 1990 deadline for paper submission of evaluation forms would be moved to coincide with the electronic submission deadline of 9 February 1990.

Tricia Oberndorf then spoke on behalf of the Requirements Subgroup requesting that if anyone noticed problems with the Requirements Document or had observations on it, please let the Requirements Subgroup know.

CDR Barbour then provided wrap-up comments, first restating that the next meeting will be held from 6-8 March 1990 at NSWC, White Oak, Maryland. He then presented information on the OS Prototype in response to numerous requests during the week. A CBD announcement was made for Ada Programming Systems on 10 January 1990, and an industry briefing is expected some time in April 1990. For more details, contact Kar Chan (SPAWAR) at (703)602-9207. Finally, CDR Barbour presented his wrap-up slide, noting that all the meetings objectives had been achieved. The 9 February 1990 due date for evaluations was reinforced. It was noted that everything possible was being done to get the documentation out to the evaluators. Contact CDR Barbour if the deadline is approaching and the documentation has not yet been received. CDR Barbour then thanked the candidate presenters for their efforts and the evaluators for their response and the meeting was closed.

NEXT GENERATION COMPUTER RESOURCES PROGRAM

OPERATING SYSTEMS STANDARDS WORKING GROUP

MEETING 6-8 MARCH 1990

MEETING MINUTES

The ninth meeting of Next Generation Computer Resources (NGCR) Program Operating Systems Standards Working Group (OSSWG) was held March 6-8 at the Naval Surface Warfare Center (NSWC), White Oak, Maryland. Approximately 40 representatives of government, industry and academia gathered to hear opening remarks Tuesday morning. The current status of the Operating System Interface evaluation was presented Tuesday morning along with preliminary results. The remainder of the day was devoted to an analysis discussion of the preliminary data. On Wednesday and early morning Thursday, the Subgroups met and discussed the reports to be written after the evaluation is completed. The meeting concluded on Thursday morning with Subgroup reports, a wrap-up, and further discussion on interpreting the evaluation results.

Tuesday, 6 March.

CDR Rick Barbour provided the opening remarks. He first emphasized to the group that all data to be presented at the meeting is preliminary and anonymous. The data is only being used as examples for discussion purposes. CDR Barbour then presented the agenda and objectives of the meeting. He stated that the main objectives were to arrive at a consensus on the evaluation analysis details, develop an approach to the recommendations, and create report outlines. CDR Barbour announced that Jim Oblinger will be the Available Technology Subgroup Chairman, as Phil Hwang will be unable to continue in this role. It was also announced that copies of the NGCR Co-Chairs' Open Issues were being distributed to help develop solutions to interoperability issues between the NGCR standards.

Jim Oblinger opened the Subgroup reports with the Available Technology Subgroup status. He stated that the subgroup had worked on and was about to finalize the Available Technology report within a week. Copies were made available for anyone to review and provide comments, after which the report would be published. Jim noted that it was important to discuss the issue of selecting a single candidate baseline versus some set of candidates as a baseline, including a discussion of possible groupings of the candidates. He then stated that the Available Technology Subgroup, along with the Requirements Subgroup, would be putting together an After Action Report outline at this meeting.

Steve Howell then presented the Approach Subgroup status. The software to perform the data analysis on the evaluation numbers had been updated. This software was used to prepare preliminary results that will be presented at this meeting. In addition, he stated that the subgroup will be presenting the current evaluation status and a revised schedule of important events. Steve then noted that the goal of the Approach Subgroup at the current meeting would be to prepare outlines for the Evaluation Results Report and the Recommendation Report.

The Requirements Subgroup status was presented by Dan Juttelstad. He stated that for part of the meeting the subgroup would be getting together with the Available Technology Subgroup to develop the After Action Report outline. Additionally, the subgroup would work on the Operational Concept Document (OCD) which is due on 1 May 1990.

Carl Schmiedekamp then presented the preliminary data. He first noted that evaluations are still arriving and that the data presented only reflects the status of the evaluation as of 1 March 1990. To avoid influencing the evaluation results, the seven operating system candidate's names were arbitrarily replaced by days of the week to allow for the analysis discussion. Carl reviewed the scoring algorithm that is being used, including the use of Weight Sets 1 and 2. He then discussed the error evaluation that is being performed. There are two types of error estimates being calculated. One estimate is obtained by using the standard deviation of scores within a particular criterion. The other estimate uses the confidence factors of each criterion. The error presented in the preliminary data analysis was chosen as the larger of the two estimates. Carl noted that the error for a criterion decreases as the number of evaluations increases. He then showed some charts of the data including a sample candidate's scores in all service classes, a sample candidate's individual criterion scores for Service Classes 0 and 1, and all seven candidate's weighted scores charted against each other for every Service Class.

Next, Carl presented the Representative Application Domains (RADs) preliminary results which were calculated using Weight Set 2. Carl noted that the RADs do not appear to reveal very much since there is little variance between candidates for any single RAD. As a result of this, an unweighted RAD was introduced, called Glass, to provide a test case. The new RAD had similar results to the other weighted RADs. The discussion of this was deferred until later in the meeting.

Carl concluded by noting that a careful audit trail of the evaluation forms was being maintained. He stated that as of 1 March 1990, there was an average of 6.3 evaluations per service class. This is short of the desired minimum of 7 responses for each service class.

Tom Conrad then presented the evaluation process status. He showed the breakdown of evaluation responses in various ways including by individual evaluator, by industry breakdown, and by number of responses per candidate per Service Class. In the latter analysis, it was evident that the response was relatively low when enforcing the requirement that an evaluator's scores would not count unless all seven candidates were evaluated in a given Service Class.

Tom then reviewed the process of scoring that was used to obtain the preliminary results and discussed the subsequent actions. The immediate concern is to collect the straggler data.

He noted that two complete sets of documentation were being made available at the meeting for evaluation purposes. To guide evaluator's in their efforts to finish quickly, Tom suggested that each Service Class should be evaluated for all seven candidates before the next class is started. Additionally, he urged that the remaining evaluators put their efforts into the needed Service Classes first (those with lower response).

Separate from the need to finish the evaluations, it was identified that a consensus on the recommendation process was needed. The decision must be made whether to choose one candidate or multiple candidates (e.g., SRAX-LPOS-KERNEL). Tom identified the significant upcoming dates:

8 March	Consensus reached on evaluation details and outlines completed for the Evaluation Results Report, the Recommendation Report, and the After Action Report.
26 March	Complete the automated analysis of the data.
6 April	First drafts of reports due.
17-19 April	SEI meeting to discuss the reports and the results.
30 April	Deliver the OSSWG recommendation to SPAWAR 324.

The reports to be written will provide the following general information:

Evaluation Process Report	-	What we did
Evaluation Results Report	-	What we found
Recommendation Report	-	What we concluded
After Action Report	-	What else must be done.

A discussion on how to analyze the data and what to do to get results was initiated by Tricia Oberndorf. She noted that there are many different views that could be taken to analyze the data. One option would be to look at which candidate is the best in the most Service Classes. If this were to be done, the Thursday candidate, which had the highest score in 8 service classes, would be considered the best baseline candidate. A problem arises however, in how to deal with the error. Some of the scores have large variances which could change the results significantly. It was suggested that a statistician's help might be needed. Tricia also noted that the programmatic issues will be analyzed separately from the technical analysis.

A second approach that could be used is to choose a passing level at which candidates would be considered acceptable and then grade from there. An example of this would be to draw a line at "6.5" and look at scores above this. Unfortunately, this analysis is inconclusive.

As another option, a total aggregate score could be obtained for Service Classes 2 through 16, without weights. This method would be similar to the Glass Representative Application Domain (RAD) presented by Carl Schmiedekamp. If this analysis were to be done, the top three candidates would have been Thursday, Tuesday, and Saturday.

A fourth option would be to use the RAD scores. These scores would show which candidates are well suited for a particular application domain (e.g., real-time). In looking at the preliminary results, however, the RADs tended to normalize the candidates. As a result, no conclusive information could be drawn from them. Since the RAD scores were inconclusive, three hypothetical candidates were created to test the results. Each new candidate had different attributes (e.g., security, Ada, fault tolerance) and was scored in each Service Class with either a 0 or a 10. The results of this test concurred with the preliminary evaluation results.

Some ideas as to why the RAD scores did not vary were discussed. One problem is that the application domains are generally realistic, but there are no diametrically opposing application domains. Thus, there is too much balance in the RADs. Additionally, the candidates themselves are well balanced as a result of the selection process. It could be concluded from this that different baselines for each application are not necessary.

Tricia then asked for further suggestions from the attendees on how to evaluate the data. To start the discussion off, Tricia presented the candidates scored against the "Big Six" requirements (1. REAL-TIME, 2. DISTRIBUTED, 3. HETEROGENEOUS, 4. ADA, 5. SECURE, and 6. FAULT-TOLERANT). The results of this analysis seemed to indicate that there were three candidates which tended to score at the top of each requirement. Other suggestions included looking at specific criteria and how a candidate scores on particularly important criteria and looking at the Ada Language Reference Manual and ARTEWG work as a new candidate. Tricia suggested that these and any other new ideas be discussed in the subgroup meetings.

CDR Barbour then presented and discussed an overview of the programmatic requirements. In the evaluation analysis, the programmatics were left unweighted, but were placed in a hierarchical order of importance. He noted that the group may want to weight the programmatic criteria. CDR Barbour stated that the ranking was derived in conjunction with the NGCR Program Office. It was not clear precisely how the scores would be used at this time.

Tricia next presented the scoring results of the programmatics with a weighting according to the hierarchy. Criteria 1 through 4 were given a weight of 10, criteria 5 and 6 were given a weight of 8, and criteria 7 and 8 were given a weight of 5. As a result of this, Tuesday and Saturday, which had scored

relatively high in the technical analysis, scored above the other candidates. However, Thursday, which scored very high in the technical criteria, had the lowest score in the programmatic evaluation. This makes the selection process even more complex.

A discussion then ensued regarding the characteristics of the operating system candidates. The seven candidates consist of great extremes. They range from ORKID which is a scaled down interface, to POSIX which is overly abundant, to CRONUS which is distrubuted. Also, some are kernels while others are full operating systems, with variations in between. This is presently not accounted for.

A suggestion was made by Mars Gralia to do a failure analysis. To do this, assume a candidate is chosen and then see if it would completely fail in an area where NGCR absolutely needed it. Neil Henderson further suggested that since there seem to be three top candidates, it might be useful to look at the three and then pick holes in them. Eventually, they will fall out, leaving a final baseline.

An analysis and discussion was then made by the working group comparing Service Class scores of the top three candidates (Tuesday, Thursday, and Saturday). The greatest difference in score between the three occurs in Service Class 3, Security.

An observation was made that a programmatic criterion appears to be missing. The criterion regards the candidate being part of an "open process", not just public domain. It was suggested that the candidate should be under public control for standardization. The need for an open process was discussed. It was noted that the cost of a publicly controlled Operating System Interface (OSIF) should be relatively low and that no radical changes should occur. On the negative side, however, the question of how to maintain the standard arose. Additionally, it was noted that large groups (e.g., POSIX) may take a long time to put out a standard.

CDR Barbour then noted that it is possible for the group to come to the conclusion that no candidate is satisfactory and that the Navy should lead the development of a new interface. Although this is not anticipated or desired, it is a possibility which should be considered. He stated that the Navy is looking to the OSSWG experts to help it establish what to do and where to go.

A consensus was reached on the baseline documents for each candidate. The candidates and their associated documentation are as follows:

Alpha - Alpha Operating System, Kernel Interface Specification (Part 7)

ARTX - VRTX32C User's Guide; MPV User's Guide; TNX-E User's Guide; IFX User's Guide; RTAda User's Guide; RTAda Board Support Package Developer's Guide; ARTX32 Engineering Implementation

CRONUS - User's Reference Manual; Programmer's Reference Manual; Software Design Document for the Experimental Secure Distributed Operating System Development

IRMX - iRMX Real-Time Kernel, Reference Manual; Nucleus Concepts; Networking Services System Calls; I/O System Concepts; Application Loader Concepts; Configuration Guide

Mach - Kernel Interface Manual; Real-Time manual pages; Kernel Modifications for the Implementation of the Security Policy; Trusted Mach Shell; Trusted Administrator Shell; Trusted Mach Audit Server Interface Document; C Threads

ORKID - Open Real-Time Kernel Interface Definition

POSIX - P1003.1; P1003.2; P1003.4; P1003.5; P1003.6; P1003.8

Possible groupings of the candidates were then discussed. The possible categorization of the candidates which belong in each category were as follows:

Distributed OS - Alpha, Mach, none
Real-Time Executive - ARTX, IRMX, ORKID, none
API - POSIX, none
Distributed Computing Environment - CRONUS, none

In addition to these groupings, it was stated that a SRAX-LPOS-Kernel combination could also be considered.

Tricia discussed the definition of kernel to make sure everyone agreed and was talking about the same thing. Initially, the OSSWG defined kernel as primarily responsible for process execution and interprocess communicaiton. For the purposes of the discussion at the June 1989 OSSWG meeting, an Operating System was defined as the full set of capabilities. A kernel was defined as some LPOS minimal subset of this full set of capabilities which is always required. Finally, an Ada Runtime System was defined as the ARTEWG does.

After these definitions were re-established, a discussion on the SRAX-LPOS-Kernel took place. Two suggestions were made. It was stated that the OSSWG should not standardize on a kernel that specifies a low-level set of primitives on which one builds operating systems, but must look at applications to LPOS and SRAX interfaces only (but cannot forget about LPOS to LPOS interfaces). The second suggestion was to put kernels that are a

subset of an LPOS in the LPOS (recognizing that the kernels are a subset intended to be seen by applications). From this discussion, it was suggested that the Real-Time Executive and API categories be combined.

Jim Oblinger then presented possible groupings of the candidates for the baseline. He noted that various conclusions can be made by the OSSWG. It might be decided that a single candidate meets the criteria and should be adopted. A second possibility is that one candidate plus extensions will provide the best solution. Third, more than one candidate might be selected as a baseline. Finally, it may be decided that no candidates will meet the criteria, in which case the OSSWG will create a new standard. Jim then presented some of his ideas on how to combine the candidates to provide SRAX, LPOS, and kernel capabilities. His ideas were then discussed.

The group then decided to take another look at the candidates as scored against the "Big 6" requirements. The criteria used to score the Big 6 were briefly discussed. It was stated that combining multiple candidates will add levels of difficulty in coordinating, tracking, standardizing, and modifying an OSIF. This is a good argument for choosing one candidate. However, a single candidate may be best fit for an LPOS, but miss completely for an SRAX. This also must be considered.

It was decided to look at the programmatics scores for each candidate one more time. The final discussion before breaking for the day involved looking at the top three candidates against each Service Class. This analysis would take the form of identifying each Service Class as a "must have" or "can be added". Tricia then explained the work that would be done for the next two days in the subgroup meetings.

Wednesday, 7 March.

The subgroups met for all day meetings to discuss the report outlines and writing assignments.

Thursday, 8 March.

The subgroups met in the morning to complete their discussion of the reports. The entire group then reconvened for the subgroup reports, final discussions, and wrap-up. The subgroup slides and wrap-up slides follow the minutes. CDR Barbour opened the session and called for the subgroup reports.

Rich Bergman presented the Requirements Subgroup report. The Requirements Subgroup met with the Available Technology Subgroup to discuss the After Action Report. From this meeting an outline and writing assignments had been established. The Requirements Subgroup then met on its own to concentrate on the Operational Concept Document (OCD). The beginnings of a working draft already exist. Issues were discussed and further writing assignments were made. The schedule for the OCD is as follows:

Draft OCD	-	10 April
Resolve Comments	-	19 April
Deliver OCD	-	1 May

Karen Gordon next gave the report for the Available Technology Subgroup. She noted that they worked on the After Action Report with the Requirements Subgroup and that writing assignments are due by 16 March. The subgroup also worked on the Evaluation Report which will provide summaries for the candidates. An outline was established and writing assignments were made. The drafts on the assignments are due by 23 March. The Subgroup also discussed the selection of independent (yet possibly multiple) baselines versus dependent (with multiple) candidates combined into one baseline. It was decided that independent and only one baseline should be chosen and then fill in the holes.

Tom Conrad then gave the Approach Subgroup report. He first presented the remaining evaluation milestones. The Approach Subgroup discussed the technical data and results, and discussed what to do next. They looked at the Big 6 and developed a set of weights for an extended application domain. The candidates will be rescored against this new set of weights. Tom then noted the Service Classes that need further evaluation by anyone who can. The Service Classes that need evaluating are 2, 3, 4, 6, 7, 9, and 14. He stated that all data should be in by close of business on 12 March. The Approach Subgroup discussed the Evaluation Results Report, developing an outline and writing assignments. The Recommendation Report was also discussed. Regarding this report, it was decided that it would be wrong for the Approach Subgroup to write the report since they created the method for evaluating the candidates. The Recommendation Report will be written by both the Requirements and Available Technology Subgroups. Finally, Tom said that all reports should be given to Steve Howell who will integrate them.

CDR Barbour then provided the wrap-up discussion. He noted that the actual recommendation will not be decided until the April OSSWG meeting in Pittsburgh. Therefore, all drafts should be geared towards what may be expected and the details can be filled in once the baseline is chosen. At the April meeting, the final data and results will be discussed, and a baseline recommendation WILL be selected. CDR Barbour informed the group that Mars Gralia had a very brief questionaire to be filled out by the group to determine the groups background and experience for the After Action Report.

CDR Barbour then requested that on a voluntary basis all industry and academic participants provide him with the amount of money that they had spent so far in supporting the OSSWG. The numbers will be kept confidential and are strictly for the Program Office's information.

To close the loop on the documentation that has been distributed for evaluation purposes, CDR Barbour stated that all evaluators have the responsibility to dispose of the documentation properly. A letter will be sent to all evaluators stating this, as well as a letter to the candidates thanking them for their participation.

The meeting was wrapped-up with the next meeting information and places to stay. The meeting will be held at the Software Engineering Institute (SEI) at Carnegie Mellon University in Pittsburgh, PA. To preregister for the meeting, call Stephanie Alba at her new number, (301) 951-2011, or send e-mail to her at ALBA@NADC.ARPA . Suggestions on hotels near the SEI are contained in the slides which follow the minutes. Finally, CDR Barbour noted that there appeared to be a consensus from the group that the candidates sufficiently cover the requirements, but the problem that lies ahead is to sort through the data and arrive at a baseline decision.

NEXT GENERATION COMPUTER RESOURCES PROGRAM

OPERATING SYSTEMS STANDARDS WORKING GROUP

MEETING 17-19 APRIL 1990

MEETING MINUTES

The tenth meeting of Next Generation Computer Resources (NGCR) Program Operating Systems Standards Working Group (OSSWG) was held April 17-19 at the Software Engineering Institute (SEI), Pittsburgh, Pennsylvania. Approximately 60 representatives of government, industry and academia gathered to hear opening remarks Tuesday morning. The results of the Operating System Interface (OSIF) evaluation were presented on Tuesday morning. The remainder of the day was devoted to an analysis discussion of the top three candidates. On Wednesday morning, the Subgroups met and worked on writing the various evaluation reports. The OSSWG reconvened on Wednesday afternoon to further discuss the evaluation results and to vote (arrive at a consensus), by anonymous ballot, on which of the top three candidates should be the OSIF baseline. On Thursday morning, the results of the ballot were presented and the Portable Operating System Interface for Computer Environments (POSIX) was announced as the OSSWG consensus recommendation. After this, the subgroups met to finalize work on the evaluation reports. The meeting concluded on Thursday morning with the Subgroup reports and a wrap-up.

Tuesday, 17 April.

CDR Rick Barbour provided the opening remarks. After welcoming the group to Pittsburgh, CDR Barbour introduced Helen Joyce of the SEI. Helen provided administrative information and gave a brief introduction to the local area. CDR Barbour then presented the meeting agenda and the goals of the meeting. The ultimate purpose of the meeting was to arrive at a consensus recommendation to the NGCR Program Office. Slides of CDR Barbour's presentation and all subsequent presentations are contained in Enclosure 1.

Rich Bergman then initiated the subgroup reports by presenting the status of the Requirements Subgroup. The subgroup had developed the Operational Concept Document (OCD) Version 0.1, and provided inputs to the Recommendation Report, Evaluation Results Report, and the After Action Report. The plans for the April meeting were to refine the reports based on comments and the results of the consensus.

The Available Technology status report was given by Jim Oblinger. They had been developing a draft After Action Report and provided inputs to the Recommendation Report. For this meeting, the subgroup planned to work on finalizing the reports.

Tom Conrad presented the Approach Subgroup's status. The subgroup had produced and distributed report outlines, and produced a draft Evaluation Results Report. The Approach Subgroup also produced the Final Data Analyses. The final data was presented at the meeting. In addition, the subgroup planned to finalize the Evaluation Process Report and the Evaluation Results Report.

Carl Schmiedekamp then briefed the OSSWG on the evaluation data results, indicating how the evaluation scores were obtained or derived. Carl also explained how to interpret the various handouts contained in Enclosure 1. First, the Extended Representative Application Domains (ERADs) were explained. The ERAD scores were derived by incorporating the appropriate Service Class 1 (General Requirements) criteria into the RADs (Weight Set 2 plus weights for criteria in Service Class 1). This was done because Service Class 1 contained important information pertinent to each of the RADs.

The overall scores for criteria, service class, and RADs were then explained. The raw scores, consisting of one candidate's individual criterion score per evaluator, were averaged over all evaluators who graded that criterion to obtain the criterion score. The criterion scores were then weighted, using Weight Set 1, to produce a Service Class Score. The Representative Application Domain (RAD) scores were then produced by weighted sum, using Weight Set 2, over the Service Classes.

Next, Carl explained the two error calculations, Sigma and Rho, that were performed. The Sigma error represents the standard deviation of raw scores in a criterion. It is used to estimate errors in weighted sums. The Rho error is based on the confidence levels of the evaluators. Since the Rho error tended to be less than the Sigma error, the Sigma error was used in the evaluation results. In the handouts, the Sigma in a score is represented by a bar, and the actual score calculated is the center of that bar denoted by a dark, bold line.

Tom Conrad next presented the Evaluation Process Results. He first reviewed the evaluation process from the initial selection of over 100 candidates to the final data points that were reviewed by Carl Schmiedekamp. This brought the process up to its current state for the meeting.

Tom also reviewed the Navy, Industry, and Academic participation that has occurred during the evaluation process. There have been 81 Navy and 226 Non-Navy participants involved in the process in some way. Of these, 21 Navy and 27 Non-Navy participants performed the actual scored evaluation. The names of these participants along with the Service Classes they scored are included in Enclosure 1.

The technical Service Classes, 2 through 16, were then reviewed one at a time. In each Service Class, the highest three or four scoring candidates were noted. Tom next discussed Service Class 1, criterion by criterion, and then presented an average score of Service Class 1 for each candidate. It was noted that the standard deviation (Sigma) goes down when a greater number of evaluators scored a Service Class, or when a Service Class has a greater number of criteria. From the analysis of the Service Class scores, it was apparent that there was no clear winner, given the large standard deviations in the scores.

Each of the criteria in Service Class 0 were then presented along with an average score for each candidate. Following this analysis, Tom presented the RAD scores. It was noted that the Glass RAD was added for analysis purposes and uses a unary weight set. Alpha scored the highest in all RADs followed by iRMX and POSIX.

In analyzing the RAD scores, it is apparent that each candidate's score is relatively the same when the error is accounted for. It was noted, with some surprise, that each of the

candidate's scores matched very closely with the Glass RAD. In particular, the Ruby RAD was identified to be somewhat of a discriminator compared to the other RADs when analyzing the weights. Yet, the scores did not support this.

Tricia Oberndorf provided some reasons as to why the RADs did not stand out. One possible explanation for this is that the RADs represent real-world applications where the requirements vary widely across many criteria. As a result, the RAD scores tended to balance each other out. Another possibility is due to the balance of strengths in each of the candidates. No single candidate was overly concentrated in one area.

Tricia then led a discussion analyzing the results of the evaluation. The technical results were discussed first by reviewing the service class scores. Particular attention was paid to the top scoring candidates in each service class. From this analysis, it was clear that Alpha, iRMX, and POSIX consistently appear at the top. Concern was raised, however, regarding a few of the low scores (sixth place) which POSIX received.

A discussion of the error (Sigmas) was then initiated. There were a number of ways to consider the error in evaluating the results. It was noted that a Sigma of greater than 3.16 in a score was significant because this is the standard deviation that would be expected from a random number generator. For example, the scores for Alpha in Service Class 6 (File Interfaces) showed 20 criteria with Sigmas greater than 3.16. This indicates that there is confusion in the understanding of the Alpha file system.

At the request of the group, Tricia presented the Sigmas for each of the essential criteria. Essential criteria are defined as those requirements which correlate directly to one of the "Big 6" requirements (real-time, heterogeneous, fault tolerant, Ada, distributed, and secure). Some of the large standard deviations are due to either a small number of evaluators or a small number of criteria in a Service Class.

It became clear that some of the differences in scores were not significant given the large Sigmas. This brought up the question of which scores are statistically significant. To solve this problem, Tricia had consulted with a statistician at NADC. The statistician verified that the numbers already generated were valid, and introduced the analysis of variance. This analysis compares two scores and their standard deviations to determine if the difference in the scores is significant. Tricia had performed this analysis on a few of the Service Classes which she presented. The group requested that the analysis be performed on the remainder of the Service Classes to ensure that the differences in scores among all the candidates were significant, and that the three candidates being discussed were truly the top three.

Next, the Extended Representative Application Domains (ERADs) were discussed. It was noted that the relative scores for the ERADs were similar to the RADs, with Alpha, iRMX, and POSIX the top three.

Another view of the service classes was then presented. This analysis identified the candidates who scored greater than 7 (commonly considered a "passing" grade) in a given service class. Particular attention was given to the top three candidates. In addition, service classes were identified when one of the top three candidates scored less than 5 (often a "failing" grade), noting that there may be some concern for that candidate in the particular

area. Service classes that did not pose a concern were identified, since these services were relatively easy to add to an interface.

It became evident from this analysis that each of the top three candidates would not fulfill all of the needs of the NGCR OSIF Standard as submitted for evaluation. One area in particular that did poorly across all the candidates is fault-tolerance. This is indicative of the fact that fault-tolerance has traditionally not been a concern in the industry.

An in-depth comparison of the scores for each of the top three candidates in each criterion was performed. In this analysis, potential "show-stoppers" were identified (those with scores less than 5 on Essential criteria). Some debate arose over the definition of show-stopper, as well as questions as to whether certain criteria were really "show-stoppers". The full analysis was deferred until the results of the analysis of variance were complete. However, it was noted that security capabilities were a major concern for all three candidates.

Tricia noted that in performing and preparing the analyses, one additional cross-check was done. Tricia looked at the raw evaluator scores and performed a relative ranking of the top three candidates. From this, an individual "vote" for one of the candidates was derived for each evaluator. The results of this informal analysis was consistent with the numbers that have already been presented.

Tricia then asked the OSSWG if anyone believed that the three candidates (Alpha, iRMX, and POSIX) were not actually the top ones to consider. After some discussion, an opinion was brought forth that the evaluation may have been too subjective. It was noted, however, that the process being performed here is very similar to that done on a large proposal evaluation. One advantage that the OSSWG has is the unusually high level of technical expertise gathered to perform the evaluation. Therefore, the subjectivity is actually technical subjectivity and not personal opinion.

The technical analysis was wrapped-up by noting that the large standard deviations indicate possible misconceptions in the candidates, possible problems with the documentation, or perhaps varied interpretations of the requirements. The OSSWG was attempting to arrive at an unambiguous answer when one may not exist for the selection of a baseline recommendation.

The discussion then turned to the programmatic issues. The results of the weighted average scoring placed POSIX and iRMX on the top, and Alpha scored at the bottom. Each of the criteria were then analyzed individually.

It was suggested that commercial acceptance may not be a good measure of the candidates. The reason for this is that commercial acceptance in 1990 probably reflects 1980 technology. Waiting for acceptance could make the technology old. Upon further discussion, however, the general consensus among those who had evaluated this criterion was that they had accounted for acceptance now as well as that projected in the future.

A discussion then insued on what capabilities were actually provided with each of the top three candidates. It was noted that if POSIX were chosen, then the other systems would

still be available as implementations of POSIX (e.g., Alpha intends to be POSIX compliant) that could also provide additional features. The danger in this is that a non-portable system could be built if the additional features were used. This led to a discussion on subsets and profiles of the standard.

One idea put forward was to copy what the SAFENET Working Group had done and adopt different standards into one Navy standard to cover all aspects of the OSIF. From this, profiles or subsets of the standard would then have to be chosen for each application. It was noted that the group may be discussing two mutually exclusive goals. One goal is to have the standard be all inclusive for every application. The other goal is to choose a standard that will apply to a dominant subset of applications. The suggestion was made to look at the SAFENET and Backplane Working Groups for lessons learned.

A concern was raised that if POSIX were selected as the baseline, could the Navy actually influence the standard at this point. It was noted that any group or individual could affect a standard provided they are well organized and come with written proposals to the group. This tended to be the majority opinion among the OSSWG members with many parallels drawn to standardization groups in the past (e.g., 802 and XTP).

Following the programmatic discussion, a "final" number was presented for each candidate. Tricia noted that this number was being presented because most people instinctively like to see an overall score, however, its significance is debateable. The score was derived by taking the weighted programmatic score and multiplying it by the Glass ERAD (technical) score. From this analysis, the top three candidates remained Alpha, iRMX, and POSIX.

Wednesday, 18 April.

The subgroups met in the morning to discuss and write the evaluation reports. The agenda was amended as a result of the previous evening's Executive Meeting and the full OSSWG was reconvened in the afternoon to further discuss the evaluation results.

Tricia first presented the results of the analysis of variance. When performing the analysis on all seven candidates, the previous results and conclusions were supported; Alpha, iRMX, and POSIX are the top three candidates.

The requirements that show a statistically significant difference between candidates were then presented and discussed. In this analysis, the "show-stoppers" that resulted from non-significant results or non-essential requirements were removed. The results of this showed, for the technical requirements, that POSIX led the three candidates with 23 "passing" grades (> 7) and 24 "show-stoppers". Alpha was second with 19 passing grades and 26 show-stoppers, while iRMX had 6 and 40 respectively.

The definition of a show-stopper was briefly discussed once again. It was determined that further discussion on the topic of show-stoppers was not warranted since a precise definition did not exist. Additionally, it was noted that it is not apparent that there really are any show-stoppers.

Rich Bergman then presented the results of the Requirements Subgroup's morning session. The subgroup had identified the strengths, weaknesses, and risks of each of the top three candidates. This presentation is contained in Enclosure 1.

An issue was raised regarding a discussion from the last OSSWG meeting in White Oak on March 6-8. This discussion revolved around the combining/layering of the candidates to establish a baseline. The group decided against such an option at that time. A suggestion was then made to reconsider this sort of option with, for example, POSIX as the generic baseline and Alpha as the standard for the SRAX (distribution). Some of the group considered this as a viable alternative, but the overall consensus of the group was that a single baseline should be chosen while using ideas from the other candidates, as appropriate, to fill in the holes.

A 45 minute period was then alloted for the group to discuss each of the top three candidates individually. These periods were strictly timed and comparisons between candidates were not allowed. Alpha was discussed first, followed by iRMX and then POSIX.

Alpha Advocacy.

The point was raised that Alpha has the only realistic approach to real-time systems. Since real-time appears to be the most important requirement, Alpha should be considered the best candidate for the OSSWG baseline.

Dr. Karen Gordon then presented her thoughts regarding Alpha. She noted that the purpose was to standardize on an interface between the operating system and the applications. Since the Alpha candidate that was submitted is only the Alpha kernel, she stated that it falls short in meeting the requirements of the OSSWG baseline. Additionally, after discussing a number of Alpha's strengths and weaknesses, Karen proposed that the weaknesses be changed to risks.

It was noted that one of Alpha's strengths is extensibility. An opinion was stated that if this is a strength, then by definition, Alpha cannot be faulted for having weaknesses in an area because Alpha should ultimately be able to be expanded to handle any weakness.

A suggestion to consider as a risk the fact that Alpha is only synchronous was made. Additionally, it was suggested that object orientation be removed as a stength since this could be extended to every one of the candidates. After some discussion, the group decided not to change these issues in the strengths, weaknesses, and risks.

Doug Jensen noted that Alpha was submitted as a kernel and uses object orientation as a model. However, it can also be looked at functionally and not just from an object oriented point of view. Additionally, Alpha does not fully specify the interface. This can be viewed in two ways. On one hand, it allows flexibility, making Alpha more open. On the other hand, the interface is not as clear if it is not specified directly.

It was further noted that many of the functions of Alpha are not readily visible in the kernel, making it hard to evaluate. For an LPOS candidate (Application Program Interface), Alpha does not appear to provide the necessary functions since it was not designed to do this.

iRMX Advocacy.

The session started by looking at the strengths and weaknesses of iRMX. It was suggested that iRMX is extensible and that extensibility be added as a strength. As a candidate, it includes all the parts of the operating system. The group voted in favor of adding extensibility as a strength.

The question was raised if iRMX plus POSIX could be considered as a candidate in the same way as Alpha and POSIX. Though the two candidates are based on different models, it was determined that the two candidates contain many overlapping services and would not make a good combination.

It was noted that one of the important requirements for a candidate is to be compatible with Ada. As a result, the conceptual compatibility of iRMX with Ada was discussed. The general consensus was that the necessary support for Ada is contained in iRMX.

The discussion then turned to the process of initiating a new standardization effort. It was stated that it would not be easy to create a new standard that is close to an already existing standardization effort (i.e., POSIX). It was noted that when initiating a standard, the IEEE works by consensus. If the support from users and suppliers exists, then a standard is typically allowed. For example, POSIX had close to 300 supporters at the outset of the standard. The fact that there already is an operating system interface standard should not interfere with a new standard. It was also noted that iRMX has a very strong user group.

POSIX Advocacy.

A suggestion was made to add the uncertainty of the schedule for the POSIX documents, other than 1003.1, as a risk. This led into a discussion of the document baseline and the scheduled milestones for each of these documents. The current schedule was noted as follows:

1003.1 (POSIX) - Published
1003.2 (Shell and Utility Application Interface) - in balloting
1003.4 (Realtime Extension) - in balloting (without threads)
1003.5 (Ada Binding) - balloting in Summer 1990
1003.6 (Security Interface) - drafts only, no ballots scheduled
1003.8 (Transparent File Access Interface for Networked Computer Environments) - balloting by end of 1990

It was noted that changes to the 1003.1, 1003.2, and 1003.4 documents are highly unlikely at this time. However, additions to these documents can be made.

One downside of POSIX being an interface and not a product is that some features do not get standardized. This occurs because the participants cannot agree on all possible implementations for the feature and do not want to make the standard limiting. As a result,

the feature is left out of the standard. In POSIX, the malloc (memory allocation) function is an example of this.

In looking at the current status of POSIX, it can be seen that there will be holes in the OSSWG requirements that need to be filled in. This can be done in one of three ways: change/add to POSIX, find other standards, or make a Navy unique standard. It was noted that there is precedence for all three of these in the Navy already.

Questions were raised regarding the coordination and integration of the "dot" documents. The group was concerned about compatibilities between sections as well as possible flaws in one document affecting another. It was noted that the overall POSIX group works together to keep all of the "dot" documents compatible. This in fact is a strong goal of the POSIX group and any incompatible parts of a document must be fixed before it is accepted.

The inclusion of real-time as a weakness of POSIX was then discussed. It was noted that real-time systems are currently being built with Unix. However, it was also noted that showing existence does not prove a strength or weakness. After the discussion, a vote was taken to remove real-time as a weakness. The vote failed and real-time remained on the weaknesses list.

After the three candidates advocacy was completed, the OSSWG discussed the general strengths, weaknesses, and risks. It was noted that Ada seems to be a problem overall. The high variances in the scores imply that the knowledge of the OSSWG was not sufficient to address Ada. It was agreed that none of the systems support Ada runtime semantics. Support for Ada runtime semantics was added to the General Weaknesses list after a vote.

Distribution performance was then discussed. It was generally agreed that this is a problem with current technology, although Alpha and iRMX are beginning to address it. The group voted to remove distribution performance from the risks list and noted that it should be discussed in the After Action report.

The issue of combining POSIX and Alpha was readdressed. It was noted that this might be more appropriate as a next step after the baseline is established. Once a baseline is chosen, the gaps will be identified and possibilities for filling those gaps will be discussed. On the other hand, it was stated that the group might want to take advantage of the combination immediately and make a statement by choosing the two candidates together. The general consensus, however, was that choosing one baseline candidate now does not preclude getting two (or more) candidates later.

Tom Conrad then displayed the RAD scores of the seven candidates, and urged the group to recall all the data that had been presented for the baseline decision to be made.

CDR Barbour introduced the process that would be used to derive a definitive consensus and select a single baseline candidate for recommendation to the NGCR Program Office. He noted that the consensus by anonymous ballot would make the selection. All OSSWG members were eligible to vote, but only evaluator votes would count for the

recommendation. Other votes would be used as advisory information. To be chosen, a candidate had to receive at least 50 percent of the vote.

Envelopes and paper were then distributed to the group and they were asked to select a single candidate from the top three: Alpha, iRMX, or POSIX. The votes were collected and it was announced that the results would be provided the following morning.

Thursday, 19 April.

The OSSWG reconvened in the morning and the results of the previous day's ballot were presented. Alpha had 12 votes that counted and 3 advisory votes, iRMX had 5 votes that counted and 1 advisory vote, and POSIX had 18 votes that counted and 6 advisory votes. Since POSIX received over 50 percent of the votes that counted, CDR Barbour announced that POSIX will be the official recommendation that the OSSWG forwards to the Program Office. CDR Barbour opened the floor for comments. No comments were made.

The OSSWG broke into subgroups to develop the final details of the reports now that a baseline recommendation had been selected. Late in the morning, the OSSWG reconvened for subgroup reports and the final wrap-up.

Rich Bergman presented the Requirements Subgroup report. The subgroup had completed the draft version of the Recommendation Report and would be distributing it for comments via ARPANET. Rich noted that the group would also continue to work on the Operational Concept Document (OCD), targeting the end of May for delivery to the Program Office. Finally, Rich identified the subgroup's plans for the next meeting. The Requirements Subgroup will begin reviewing the OCD Model and Requirements against POSIX, as well as refine the OCD for its next release.

Jim Oblinger presented the Available Technology Subgroup report. Significant updates were made to the After Action report and Jim outlined the process to incorporate these and solicit comments via ARPANET. The subgroup had also generated significant inputs to the Evaluation Results Report. For the next meeting, Jim stated that the Available Technology Subgroup will begin to draw the requirements out of POSIX to identify holes. The subgroup will also work to find ways to fill in the holes.

Tom Conrad then presented the Approach Subgroup report. A walk-through of the Evaluation Results report had been completed and Tom identified the plans to complete the report and obtain comments, also via ARPANET. Tom noted that the OSSWG is now going through changes based on the selection of a baseline. As a result, the subgroup will be developing a new POA&M during the next meeting. Additionally, they will be arranging a briefing of the results of the "Bidder's Conference" on the Operating System Prototype Contracts. The Bidder's Conference will be held on 23 May 1990 at NUSC in Newport, Rhode Island. Tom also noted that since no bias was found in the scoring from the evaluators, it would not be discussed in the Evaluation Results report.

CDR Barbour then provided the meeting wrap-up. He first noted the next meeting which is to be held on June 5-7 at NSWC, White Oak, Maryland. CDR Barbour stated that the objectives of the meeting had been accomplished and the OSSWG is on target. Copies

of the final reports will be distributed to all OSSWG members. Additionally, letters will be sent to each evaluator identifying the proper steps to close out the evaluation (i.e., handling of documentation).

CDR Barbour then thanked all of the members for participating and invited them to continue participating in the future. He noted that schedule changes are anticipated as a result of the decision made and the changing focus of the OSSWG. The group was then congratulated on the tremendous effort and results that were achieved. CDR Barbour then closed the meeting.

NEXT GENERATION COMPUTER RESOURCES PROGRAM

OPERATING SYSTEMS STANDARDS WORKING GROUP

MEETING 5-7 JUNE 1990

MEETING MINUTES

The eleventh meeting of Next Generation Computer Resources (NGCR) Program Operating Systems Standards Working Group (OSSWG) was held June 5-7 at the Naval Surface Warfare Center (NSWC) in White Oak, Maryland. Approximately 45 representatives of government, industry and academia gathered to hear opening remarks Tuesday morning. The Operating System Prototype (OSPROT) Industry brief and the response from it was presented on Tuesday morning. The latter part of the morning was devoted to strategizing for the OSSWG to attend the POSIX meetings. In the afternoon on Tuesday information on the NIST POSIX Conformance Test Suite was presented and then the OSSWG discussed the Delta Document which will identify the differences between the Requirements Document and POSIX. The subgroups then met for the remainder of the day and all day on Wednesday. On Thursday morning, the OSSWG broke into OSSWG/POSIX extension groups to organize and strategize for the July POSIX meeting. The meeting concluded on Thursday morning with the OSSWG/POSIX extension group and Subgroup reports and a wrap-up.

Tuesday, 5 June.

CDR Rick Barbour provided the opening remarks. After welcoming the group to White Oak, CDR Barbour provided brief administrative information and presented the meeting agenda. He then announced POSIX as the official Navy Operating System Interface (OSIF) baseline selection. The goals of this meeting were to strategize on the next phase of the OSSWG. This phase, OSSWG Phase II, will involve identifying the differences between POSIX and the OSSWG Requirements document and then influencing POSIX as much as possible to incorporate these differences. Slides of CDR Barbour's presentation and all subsequent presentations are contained in Enclosure 2.

Rich Bergman then initiated the subgroup reports by presenting the status of the Requirements Subgroup. Since the last meeting, the subgroup has provided inputs to finalize the evaluation reports, completed the Recommendation Report, and has issued the Operational Concept Document (OCD) version 0.1A for comments. At the current meeting the Requirements Subgroup planned to continue developing the OCD towards a July delivery to SPAWAR.

Jim Oblinger provided the Available Technology Subgroup status report. The group had completed the After Action Report which will be released on 15 June 1990 as NUSC TD 6904, dated 1 June 1990. They also worked on producing the Available Technology Report as a TD and attended the OSPROT Industry brief. The group planned to start working on the POSIX to OCD Delta Document and focus on OSSWG to OSPROT communication at this meeting.

Tom Conrad then presented the Approach Subgroup status. The subgroup had finalized the Evaluation Process and Evaluation Results Reports and participated in the OSPROT Industry brief. They intended to draft a new OSSWG Plan of Actions and Milestones

(POA&M) for this meeting. Tom then identified the key items that the OSSWG needed to start considering and working on at the meeting. These items are:

- Finalize the 5 reports (Recommendation, After Action, Evaluation Process, Evaluation Results, and Available Technology)
- Identify the POSIX baseline (which "dots" are included)
- Finalize the OCD
- Start work on the Delta Document
- Establish a strategy for moving the POSIX baseline to meet the OCD requirements
- Define the OSSWG support role with respect to the OSPROT
- Define a Conformance Testing methodology
- Create a new POA&M and restructure the OSSWG to support it.

Tom presented the organization of the OSSWG as it had been structured for the baseline selection. He noted that the organization of the OSSWG would need to change in the next phase to support the goals listed above. This would be worked on by the Approach Subgroup at this meeting.

Kar Chan presented the OSIF Evaluation Model Industry Briefing and discussed what occurred at that briefing given on 23 May 1990. The briefing is contained in Enclosure 2. Part of the discussion involved the contract structure flow for the OSIF implementation between the OSSWG and OSPROT. Kar stated that the OSSWG will deal with creating and updating the requirements and the OSIF standard. The OSPROT will take the standard and any changes identified by the OSSWG and enforce these upon the implementation. The OSPROT will also feedback to the OSSWG any issues that arise out of the implementation. It was noted that the contract structure identified Backplane inputs but did not identify the LAN working group. Kar stated that this will be corrected.

The industry representatives present at the 23 May brief had some concern over Ada requirements. They asked if other languages could be used in addition to Ada, or perhaps not even use Ada. The concern revolved around the fact that there may not be any Ada operating systems that are POSIX compliant.

CDR Barbour then discussed Phase II of the OSSWG. He presented an updated schedule for the OSSWG which includes attending the POSIX meetings. One concern about the schedule is that POSIX will be balloting on a number of "dot" documents in the near-term and the OSSWG must act quickly to get involved. ("Dot

refers to the various POSIX extensions being developed to the standard p1003.1. In this context, "dot 4" refers to the p1003.4 extension for real-time services). CDR Barbour identified the near term OSSWG milestones to write the Delta Document, refine the Abstract Model, and finalize the OCD by December.

It was noted that there has been very positive feedback from the Navy on the work coming out of the OSSWG. CDR Barbour stated that he had inquired about IEEE memberships for OSSWG members participating in POSIX. The Navy will not pay for the memberships. To vote in IEEE ballots, you must be a member of either the IEEE or the IEEE Computer Society. You need not be a member of both. The Computer Society membership costs about $40 and the IEEE membership costs about $100. (By the way, we gain access to documents via our subscriptions.) CDR Barbour noted that the POSIX subscription costs were being investigated.

Tricia Oberndorf next discussed the POSIX committees and the OSSWG strategy. She presented all the current POSIX committees and identified the specific groups of interest to the OSSWG. The P1003.3 committee had been handling all test methods for every dot group and will probably break out its responsibilities to each dot group. The P1003.8 committee was responsible for all network services, but the network services have now been divided into a few dot groups. Tricia also noted that the Technical Committee on Operating Systems (TCOS) had withdrawn its support for P1003.13, Namespace and Directory Services.

Tricia stated that although P1003.1 is the key to POSIX, it is already published and is not likely to change at this time. As a result, the OSSWG will concentrate its efforts in the other groups. It was noted that if pressure from the ISO and European communities cause changes in P1003.1, then the OSSWG will probably want to participate at that time.

Tricia then discussed related efforts to POSIX by the IEEE. These groups include P1201.1, P1201.2, P1201.3, and P1201.4 which are not of interest to the OSSWG. The P1237 committee on Remote Procedure Calls (RPC), which has no current draft and is targeting a Summer 1992 standard, is important to the OSSWG. Also, P1238.1 (OSI API) and P1238.? (FTAM API) are important to the OSSWG. Finally, Tricia suggested that a liason from the OSSWG should attend the Control and Status Register (CSR) committee, P1212.

One area of major concern with POSIX is fault-tolerance. Currently, there is no working group addressing fault-tolerance. This is an important area for the OSSWG and a strategy to address it must be determined.

Tricia then discussed efforts by the IEEE that are related to POSIX. The five areas that are most important to the OSSWG are:

- System Interface Work (real-time)
- Distribution Services

- Security
- Conformance Testing
- Ada Binding

The OSSWG, it was stated, will need to divide into these groups based on indicated interests of the members. Preferences for POSIX committees were then collected by Tricia. She noted to the group that, although POSIX tends to be dominated by vendors, the OSSWG representatives (system users) at POSIX will have the advantage of a well defined set of requirements.

Jim Hall of the National Institute of Standards and Technology (NIST) briefed the NIST POSIX Conformance Test Suite for FIPS 151-1. He first discussed FIPS in general and then keyed on the POSIX conformace suite. Jim's presentation is contained in Enclosure 2.

Next, Tricia discussed the Delta Document. The objectives of the document are to form a basis for identifying enhancements to POSIX and to use it as an inclusion in the OSPROT RFP. In identifying the gaps between POSIX and the OCD, one of several options can be considered for each requirement not fulfilled, including:

- Requirement is unnecessary and can be discarded
- Requirement is fulfilled by SAFENET
- Requirement was previously considered and discarded by POSIX
- Requirement is nice to have, but not really needed or worth working toward
- Requirement is "too far out" and it would be premature to standardize at this time
- Requirement is a must ("got to have") and must be included even if POSIX does not.

From the list of requirements being pursued, an approach to take them into POSIX must be determined explaining the concepts, rationale, and interfaces required.

A question was raised about handling conflicts with POSIX. Tricia responded that if a necessary requirement conflicts with POSIX, then the OSSWG will have to diverge from POSIX for this or identify a profile around the problem.

Tricia emphasized that a final Delta Document is needed by December 1990. This is an important deadline for meeting the OSSWG's own needs. This is also important since the document will be included in the OSPROT RFP expected around that time.

The point was raised that the POSIX committees often consider whether an efficient implementation can be made before they determine if it's worth standardizing on the interface. This led to a discussion on performance issues. It was noted that the OSSWG does not want to ignore performance, but it could not be the major focus for the standard; it is a suitable topic for a companion document. The Representative Application Domains (RADs) were identified as the best area for the OSSWG to identify performance requirements. It was suggested that they could be expanded to include performance issues.

For the remainder of the afternoon, the three OSSWG subgroups met. The Requirements Subgroup concentrated on the OCD, the Available Technology Subgroup concentrated on the Delta Document, and the Approach Subgroup concentrated on the POA&M.

Wednesday, 6 June.

The OSSWG worked in the subgroups for the entire day.

Thursday, 7 June.

The OSSWG broke into five STAR groups: System Interface Work (real-time), Distribution Services, Security, Ada Binding, and Guide. (The conformance test group identified by Tricia on Tuesday did not meet); the Guide (.0) was overlooked on Tuesday. Each group was to identify a leader, the committees which each member would attend, and then strategize on the best way to approach the POSIX committee in July.

CDR Barbour reconvened the OSSWG after the POSIX groups met. He suggested that each group leader contact the POSIX committee chair in preparation for the July meeting. He then initiated reports from each of the POSIX groups by reporting on the Guide (.0) group. They will be involved with interface issues and how POSIX will work together. CDR Barbour will take the lead in this group.

Jim Leathrum reported for the Ada group. He noted that he will take the lead for the group until August. They want to coordinate closely with the P1003.4 (Real-time) committee and push P1003.5 (Ada binding) in that direction. They also want to influence P1003.5 towards integrating with the other extensions as well.

Tony Carangelo reported for the Security group. Mark Karan will take the lead at this time. The group wanted to identify the strategy taken by the P1003.6 (Security) committee in creating the current draft standard. They then want to identify the OSSWG requirements which are not satisfied in the POSIX documentation and influence the P1003.6 committee to include these features. The group also intends to identify a set of additional requirements for Navy systems which POSIX may be able to address and then introduce these into the committee.

The Real-time group report was given by Del Swanson. He stated that the group will ask Frank Prindle to take the lead. The group noted that the real-time POSIX committees are dynamic and break into factions on issues. Their strategy will be to introduce the OSSWG requirements into the committees, develop allies, and continue to meet in the Real-time OSSWG group to strategize and share ideas. The group identified multi-nodal systems and fault-tolerance as areas not currently addressed by POSIX that they would like to introduce.

Greg Bussiere reported on the Distribution group and will be taking the lead. He noted that the group would inquire into the P1003.13 committee which was disbanded and would monitor and/or participate in P1212 CSRs. The group identified global memory management, global time services, and message transfer protocols as areas not covered within POSIX. The group hoped to keep close to the Ada and Real-time groups to share ideas.

CDR Barbour then noted that he had been in contact with groups within the Government which were interested in forming alliances. Particularly, he mentioned NASA, NIST, and the DOE. It is desirable for each group to get familiar with each other's issues and go to POSIX in unison. These groups are not as organized as the OSSWG, but they do have regular POSIX attendees.

The Subgroups then reported on their progress. Rich Bergman began with the Requirements Subgroup report. The subgroup completed a group review of the OCD version 0.1 and determined that there would be no major modifications for the July version. The only major modifications expected are to the Reference Model and these would be included in the December version. All comments to the OCD should be made to Dan Juttelstad by e-mail to RQTSG@TECR.NOSC.MIL (formerly RQTSG@.NOSC-TECR.ARPA). The subgroup planned to deliver the OCD version 0.2 to SPAWAR 324 prior to the July POSIX meeting.

Dr. Karen Gordon gave the Available Technology Subgroup report. The subgroup had generated a draft cross-reference matrix of OSSWG service classes versus POSIX committees, assembled 6 groups from the 16 service classes and assigned lieutenants to coordinate the Delta Document comments, and developed a draft form to submit those comments. They intended to investigate the submission of these comments by e-mail. They planned to obtain comments, categorize and discuss unfulfilled requirements, and prepare the Delta Document as the comments are submitted. It was noted that the Delta Document should also address holes that might arise in the OSSWG Requirements as a result of studying POSIX.

The Approach Subgroup report was given by Tom Conrad. Tom presented the OSSWG Phase Plan. Phase I has been completed and Phase II is beginning. The OSSWG needs to organize itself within a larger group (POSIX) and influence the standard to make it usable to the Navy. In addition, there are still NGCR and OSSWG items to complete.

Tom discussed the OSSWG organization for Phase II. The OSSWG breaks down into two separate areas. One area is to cover POSIX and is handled by the six functional groups which met on Thursday morning. The other area is covered within the OSSWG by three subgroups: Approach, OSIF Standard Evolution, and Product Transition. Tom then presented a schedule of the documents and important events affecting the OSSWG. The relationships between the OSSWG and the other NGCR working groups were shown. Tom noted that the boundaries between the OSSWG and SAFENET and the OSSWG and Backplane must be defined. He also emphasized that a Conformance Test Methodology will be drafted and provided as input to the NGCR Conformance Test group.

As the old OSSWG subgroups still have work on-going, Tom stated that these old subgroups would still continue in their current form at least through July 1990. The new groups will start up immediately and Tom requested that OSSWG members consider which new subgroup they would like to participate in. Additionally, the six POSIX groups will start up immediately.

CDR Barbour then provided the meeting wrap-up. He brought attention to the next meeting in Danvers, Massachusetts, 16-20 July 1990 which will occur in conjunction with the POSIX meeting. The OSSWG will meet in separate meetings from POSIX on the evenings of the 17th and 19th. CDR Barbour noted that the size of OSSWG had decreased a little and that it is important to assure a critical mass in all areas of work identified. He reiterated that the POSIX documentation procurement and distribution issues will be investigated. Additionally, CDR Barbour stated that he will work on firming up alliances with the other groups mentioned. He then thanked everyone for their hard work and participation and adjourned the meeting.

NEXT GENERATION COMPUTER RESOURCES PROGRAM

OPERATING SYSTEMS STANDARDS WORKING GROUP

MEETING 16-20 JULY 1990

MEETING MINUTES

The twelfth meeting of Next Generation Computer Resources (NGCR) Program Operating Systems Standards Working Group (OSSWG) was held in conjunction with the IEEE POSIX Working Group meetings in Danvers, MA, from 16-20 July 1990. Approximately 35 representatives of government, industry and academia met in two OSSWG evening sessions. The OSSWG meetings focused on areas in which the Navy could significantly contribute to the POSIX efforts. Updates of the work being pertormed in the POSIX working groups and information on the overall status and direction of the POSIX standardization effort were provided.

Tuesday, 17 July.

The meeting was opened by CDR Rick Barbour. He introduced the OSSWG Co-Chairs, for the benefit of any new attendees from the POSIX group, and then discussed the evening's agenda. First the OSSWG Subgroup Chairs would report on their group's status, then the "Star" groups would breakout and discuss the integration of the OSSWG into POSIX. In particular the Star groups were to address the status of their particular OSSWG requirements against POSIX features and schedules, and identify areas in which the OSSWG can become actively involved.

The OSSWG Subgroup reports began with the Requirements Subgroup presented by Rich Bergman. Version 0.2 of the Operational Concept Document (OCD) has been released for review on E-mail. Rich requested that comments be provided as the document will be finalized in December.

Jim Oblinger provided the *Available Technology Subgroup* status report. The collection of data for the Delta Document had begun and the group had received data from the evaluation reports. During the POSIX meeting, the Available Technology group intended to refine the data collection procedures among the Delta Document workers. Jim noted that all Delta Document data should be submitted by the August OSSWG meeting. At that meeting the analysis of the data will be discussed.

Steve Howell presented the Approach Subgroup status for Tom Conrad. The next generation of the POA&M is being developed and a draft should be available at the August meeting. The Evaluation Process and Evaluation Results Reports are still undergoing internal review (editing) at NSWC-WO and should be available soon. At the August meeting, the group will continue defining the direction of the OSSWG, especially given the results of the first POSIX meeting.

Following the OSSWG Subgroup reports, the Star Groups reported their initial findings based on the first two days of POSIX meetings. CDR Barbour began with the P1003.0 POSIX Guide report. The JTAP/ISO liason, Jim Isaac, had identified that the ISO community required that a language independent specification be written for the P1003.1 document. CDR Barbour noted that all upcoming PARs under POSIX must be language independent; however, current PARs will not be rewritten and it is only suggested that they provide language independence. The 1003.0 group discussed the usage of "POSIX" as an adjective or a noun. It was determined that POSIX is an adjective. This means that "POSIX" will always be used with something else, such as a POSIX AEP or a POSIX Open

System. The group also discussed and is attempting to define what POSIX is and how to create profiles. The OSSWG, along with AT&T, volunteered to provide Section 4.1 of the 1003.0 Guide. The 1003.0 group is targeting a mock ballot for January 1991.

Tricia Oberndorf then discussed the 1003.0 Guide and its purpose. It is intended to be an umbrella document for P1003 to provide guidance and consistency among the various PARs. It was identified that the guide needs a reference model, similar to the OSSWG reference model, to explain what is meant by Open Systems and Operating Systems. This will be approached by first defining services, and then discussing all standards which apply. Finally, profiles will be identified. The group defined a profile as "a walk through the standards, picking appropriate parts of each standard to build a coherent, open system." The 1003.0 group will also identify areas of the reference model which are not currently covered by the standards.

Tricia noted the profile discussions of the 1003.0 group, particularly the minimum requirements to be considered a POSIX profile. POSIX compliance will probably be defined as P1003.1 minus some part(s). POSIX compliance may also be defined as P1003.2 minus some part(s). These issues are being worked on by the group in addition to the possibility of identifying a maximum set of standards for POSIX compliance.

The need to discuss how the OSSWG schedule relates to the OSPROT at future meetings was identified. It was also noted that POSIX and the idea of profiles must begin to be introduced into the Navy along with efforts to make this a Navy Policy.

Mars Gralia next discussed the Ada Binding (P1003.5) group's initial findings. He noted that the 1003.5 work is not a run time environment, but identifies how an Ada programmer gets access to POSIX services through Ada. The group consists of 12 people, of which 6 are from the OSSWG. This gives the OSSWG a good deal of influence in the group. The 1003.5 document is expected to go to ballot in August. Mars noted that the document will be sent to the ISO community, but they are looking for language independence. As a result, they will be asked what is needed to fix it and get it approved rather than to review and approve it.

Although the 1003.5 document is going to ballot, there are a number of areas in which the OSSWG can get involved. These include ballot resolution, liason to the 1003.3 testing group, liason to the language independence group, and liason to the 1003.10 group. Mars noted that the Army CECOM has an agreement with another party to produce an Ada binding to the 1003.4 Real Time document. This might be an area of interest to the OSSWG. Finally, Mars noted that the 1003.1 document is incomplete when combined with Ada. There are a number of inconsistencies which must be identified and resolved.

Gail Holmes presented the status of the Conformance Test (P1003.3) Star Group. The group consists of approximately 30 members divided between two subgroups, P1003.3.1 and P1003.3.2. The first subgroup is involved with Test Methods for Measuring Conformance to IEEE Standard 1003.1 - 1990, and the latter is for Test Methods for Measuring Conformance to IEEE Standard 1003.2. The remainder of the conformance tests for each dot group will be written by the individual group with guidance from the 1003.3 members. Gail then discussed the NIST involvement in validating POSIX compliant

systems. They have developed a test suite (NIST - PCTS: 151-1) designed against P1003.1, version 12; however, the current version of the 1003.1 document is version 13. The second version of the test suite will be designed against 1003.1a. Other efforts related to the conformance test efforts which could be of value include the Phoenix Project and X-Open.

Gail noted that the current documents being written are well established, thus the major area of participation with these will be to get involved with the balloting efforts. The future of the conformance testing for the remaining dot groups will be decided by the Sponsor Executive Committee (SEC).

The Distribution Star Group report, which consists of 1003.8, 1003.12, 1003.13, 1237, and 1238, was given by Greg Bussiere. All the groups, with the exception 1003.8, are relatively immature. As a result, there should be a number of opportunities for the OSSWG to provide influence. The 1003.8 Transparent File Access group has two OSSWG members attending. The group intends to develop two profiles, a full profile and a core profile. They are working on draft 2 and expect to go to a mock ballot in September 1990. Comparisons to the OSSWG requirements could not be made at this time.

The 1003.12 Protocol Independent Interface group is a new group with two OSSWG attendees. This group intends to develop two documents, one in 6 months and the other in a year. The documents will be based on TCP and OSI profiling.

One member of the OSSWG attends the 1003.13 Name Space/Directory Services group. The OSSWG requirements are very broad in this area and the 1003.13 group is working at a lower level detail. The group has recently been reorganized. The document will be based on X-Open, but the schedule is unknown.

Greg noted that the 1237 group did not meet at this POSIX meeting and then reported on the final group, 1238 FTAM. This is a new group and is very much interested in the user's point of view. They have been working on two documents since April 1990. These documents are Objectives, Requirements, and Definitions, and Strawman Function Feature List. Their schedule is presently unknown.

Frank Prindle reported for the Real Time Star Group which consists of the 1003.4 and 1003.13 (formerly 1003.14) groups. The 1003.4 group is a relatively large group consisting of close to 60 participants. Based on a quick first look, Frank noted that, minus Ada, there seems to be about 50 percent compliance between the OSSWG requirements and the 1003.4 standard. The Ada issues have been discussed within the 1003.4 group, but there is a lack of interest in addressing them. They feel that their charter calls for C only. The issue may be addressed through threads. The OSSWG could be very influential in swaying attitudes and helping to address these issues.

The 1003.4 work has been divided into two parts: the basic 1003.4 document and 1003.4a P-Threads. The 1003.4 document just went to ballot. It was noted that much of the document is in conflict with existing state-of-the-art developments (e.g., X-Open). The 1003.4a document will probably go to ballot after the October meeting. The current 1003.4a document does not appear to match well with the OSSWG requirements. It was noted that the OSSWG must give good, well supported reasons when voting against a

document to maintain good relations within POSIX. The suggestion was made to try to form alliances within the group, especially with members who have previously been silent. Frank also noted that there is talk of a new 1003.4 group forming to modify and enhance the current document.

Frank then reported on the 1003.13 Real Time AEP group. There is a lot of room in this group for the OSSWG to get involved. This group's efforts are concentrating on 1003.1 and 1003.4 only. No distribution or security work is currently scheduled. The group will develop three profiles: a minimal embedded profile, a medium profile, and a full capability profile.

Mark Karan concluded the Star Group presentations with the Security group, P1003.6. The current 1003.6 document is well defined and should go to a mock ballot in January 1991 and a full ballot after the April 1991 meeting. Version 5 of the 1003.6 document had been reviewed against the OSSWG Requirements and is consistent. It was noted, however, that the document falls short of meeting all of the security requirements that the OSSWG will need. The current version (7) of the 1003.6 document will be reviewed against the OSSWG requirements.

The 1003.6 group has a strong vendor influence and the document is nearly complete. The OSSWG influence to this document is minimal. However, a liason group has been created which combines representatives of the 1003.6, 1003.7, and 1003.8 groups. This group will be discussing how to integrate security with system administration and distribution. The OSSWG can be of great influence in this area. The 1003.6 group does not intend to profile at this time.

The OSSWG meeting adjourned until Thursday.

Thursday, 19 July.

CDR Barbour opened the session and identified the agenda for the evening. CDR Barbour and Tricia Oberndorf would first discuss the relationship of POSIX to other international standardization groups. Then Fritz Schultz of NIST was scheduled to discuss the 1003.0 document followed by Star Group status reports.

Tricia Oberndorf discussed the makeup of the IEEE POSIX effort and compared it to the ISO. The IEEE falls under the ANSI as an accredited standardization organization. Under the IEEE are a number of societies, one of these being the Computer Society. The Computer Society further breaks down into technical committees. The Technical Committee on Operating Systems (TCOS) contains a Sponsor Executive Committee (SEC), as one of a number of committees, which oversees the standardization efforts including P1003, P1201, P1237, and P1238.

In describing the ISO breakdown, Tricia noted that the ISO in conjunction with the International Electro-technical Committee (IEC) have formed Joint Technical Committee 1 (JTC1). JTC1 breaks down into Topical Study Groups (TSGs), Special Committees (SCs), and then Working Groups (WGs). Internationally, POSIX (9945) falls under WG15

which is a subgroup of SC22 under TSG1. In addition to WG15, there are Rapporteur Subgroups on Security, Internationalization, and Conformance Test.

CDR Barbour next discussed the role of the U.S.TAG for SC22, WG15. It is a group of 18 liaisons headed by Don Terry of Hewlett Packard. The group represents U.S. issues and positions to the ISO. CDR Barbour then presented the status of the P1003 groups in the international arena. The 1003.1 and 1003.2 efforts correlate to the ISO 9945-1 and 9945-2 groups, respectively. The 1003.3 documents must be recirculated to WG15, while the 1003.4 effort is caught in the language independent specification issue between the IEEE and ISO groups. Security, 1003.6, will be merged into 1003.1 in the ISO arena. The 1003.7 efforts are behind, 1003.5 and 1003.9 are being sent to WG15, and the plans and schedule for 1003.0 have not been established at this time.

The status of the SEC, which oversees the standardization efforts, was then discussed. CDR Barbour noted that the SEC is responsible for standardization procedures, mailings, and revisions. The committee discussed the old 1003.13 Name Space and Directory Services PAR which has been reintroduced for approval. The PAR was accepted, noting that progress must be shown in the near future. The SEC then delayed a motion to rescind sponsorship of the P1224 X.400 Messaging PAR until January 1991. At that time the group is expected to show a draft document. Finally, the group discussed the submission of a PAR for windowing.

The Star Groups then met to discuss the current status based on the POSIX meetings. In addition, these groups were asked to identify any missing POSIX groups or any existing POSIX groups which are not already being covered by the OSSWG. Furthermore, the groups were asked to identify issues that they feel the OSSWG should consider.

Gail Holmes reported first on the Conformance Test status. The 1003.3.2 document is still being worked on while the 1003.3.1 document is on Draft 10 and is just being updated. Gail noted that the proposal to have each of the dot groups write their own test assertions is being met with some disapproval from the various groups.

Frank Prindle then reported for the Real Time group. The 1003.4 draft balloting was discussed and 3 issues were brought forth for the group to resolve. The 1003.4a group discussed possible resolutions of the p-threads issues. The Real Time group will try to make a strawman proposal (based on process management, signaling, and synchronization) by the next POSIX meeting to bring the issues out in the 1003.4a group. Frank expressed concern over the current p-threads, especially multi-processing and scheduling, and noted that they may conflict with the Navy's interests.

In the 1003.13 Real Time AEP group, the OSSWG is providing strong participation. The OSSWG participants have volunteered to identify and define three different sized profiles, small, medium, and large, based on surface, subsurface, and air examples.

Frank noted that "distribution" does not seem to be a major concentration across POSIX. In addition, concerns were raised over multiprocessing. The need for a representative on the 1003.14 Multiprocessing group was identified. Frank stated that the

OSSWG should be concerned with the language independence issue from both a political standpoint and so that Ada and other languages will meet with less resistance in the future.

The Ada status report was presented by Mars Gralia. The 1003.5 group wrote a pitch to the SIG-ADA group identifying what is in the document in order to gain their support. The group also discussed what to do after balloting to clean up the document, write test assertions, and handle the language independence issue. Mars agreed to look into writing a PAR for a 1003.4 Ada Binding by January 1991. As part of this, Mars requested that potential participants in the group let him know so he could provide a list to the 1003.5 Chair. This list of potential participants is needed for the PAR to determine the potential impact on current 1003 efforts.

Mars noted that the relationship of 1003.5 to 1003.1 is not the main concern of the OSSWG. The real concern will be over the binding of Ada to the 1003.4 efforts. He further noted that an Ada binding to 1003.6 Security is not a major concern of the POSIX community, but could be important to the OSSWG. Finally, Mars said that a 1003.5 binding to 1003.2 was being considered.

Greg Bussiere next presented the Distribution Star Group report. The 1003.8 document will go through a mock ballot in January 1991. It is not clear at this time who will receive copies to review. This will be decided by the 1003.8 Chair. The other groups falling under distribution are looking at mock ballots in the Summer of 1991. Greg noted that the 1237 group will be meeting separately and someone may be needed to attend if a Distribution group member cannot.

It was noted that the OSSWG SRAX level of the reference model may not be covered by POSIX. A proposal by Martin Marrietta to the 1003.13 group was then discussed. The proposal is for a real time distributed system and is contained in Enclosure (2) to these minutes.

The Security report was presented by Mark Karan. Most of the work on the 1003.6 document is occurring in subgroups for Discretionary Access Control (DAC), Mandatory Access Control (MAC), Audit, and Privilege. The language independence issue is a concern to the group, but no work is currently being done pending the efforts of the 1003.1 group. Mark noted that important issues regarding distribution and security will be addressed in the 1003.6, 1003.7, and 1003.8 Liason group. This should be an area of concentration for the OSSWG. It was also noted that a representative to the 1003.7 System Administration group might be needed. Additionally, some potential holes which need to be studied involve the relation of the 1003.6 document to the other dot groups (e.g., 1003.4).

CDR Barbour and Tricia Oberndorf then presented the 1003.0 Guide status report. CDR Barbour noted that the OSSWG participants will take an active role in helping to write the 1003.0 reference model which should be similar to the OSSWG reference model. Additionally, he volunteered OSSWG expertise to flesh out the Graphics section of the 1003.0 document. He is expecting the NUSC personnel doing the Graphics study to actually perform this work in conjunction with the Services section that Tricia and Carl Schmiedekamp are performing. Most of the issues in the group revolve around POSIX profiling and discussions on the use of the 1003.0 document.

Tricia noted that there is an issue over how much does the OSSWG really care that the results of its efforts become a POSIX approved profile. The benefit of this is that a commercial product may become available which could be purchased off the shelf. However, reality says that this may never come about, especially considering some of the apparently Navy unique requirements (e.g., Fault Tolerance).

Tricia raised the question of whether the current POSIX profiling efforts will be enough to satisfy the OSSWG's needs. If the gemstones are considered, this is probably not the case. Tricia looked at the Big 6 requirements against POSIX and placed the gemstones in them. She noted that Ada occurred consistently across the gemstones and that Fault Tolerance was a major weakness.

The idea of a POSIX standard profile was then discussed. The 1003.0 group noted that any standard can be used in a profile. As a result, they had developed the idea of a "corral" around standards that can be in a POSIX Standard Profile based on an Open System. With this view, profiles would get tentative approval as a POSIX profile until they could go through a balloting process. This discussion was not resolved in the 1003.0 group. Tricia noted that profiling might be an effective way for the OSSWG to bring its requirements into POSIX by identifying what Navy applications need rather than just stating what should be done. These profiles should be well developed outside of the POSIX forum before being introduced there.

Fritz Schultz was detained in a separate meeting and was unable to present the 1003.0 overview to the group. This will be rescheduled for another meeting.

Jim Oblinger noted to the group that the work on the Delta Document must be a top priority. There is a lot of work to be done before the August OSSWG meeting.

CDR Barbour then suggested that all OSSWG members sign-up through the IEEE to obtain the 1003.0 document as well as the other dot documents of interest. He suggested that everyone read and comment on the 1003.0 document, especially the profiling section. CDR Barbour then emphasized that OSSWG members should volunteer to take charge in as many areas as possible which could affect the OSSWG. He then requested that each of the Star Group leaders provide the following information:

- an attendance list of companies attending the dot groups
- what have the OSSWG participants committed to
- holes in POSIX as it relates to the OSSWG

The next meeting of the OSSWG will be held on 28-30 August 1990 at NSWC, White Oak, Maryland. All members attending this meeting should preregister with Stephanie Alba (301-951-2011 or e-mail alba@nadc.nadc.navy.mil) by 24 August 1990. CDR Barbour then noted that it was apparent that the OSSWG had made an impact on POSIX and emphasized the importance of remaining diplomatic. The meeting was then adjourned.

NEXT GENERATION COMPUTER RESOURCES PROGRAM

OPERATING SYSTEMS STANDARDS WORKING GROUP

MEETING 28 - 30 AUGUST 1990

MEETING MINUTES

The thirteenth meeting of Next Generation Computer Resources (NGCR) Program Operating Systems Standards Working Group (OSSWG) was held at the Naval Surface Warfare Center (NSWC) in White Oak, Maryland. Approximately 30 representatives of government, industry, and academia attended. The OSSWG meetings focused on areas in which the Navy could significantly contribute to the POSIX efforts. Updates of the work being performed in the POSIX working groups and information on the overall status and direction of the POSIX standardization effort were provided.

Tuesday, 28 August.

The meeting was opened by CDR Rick Barbour. He presented the agenda for the meeting and discussed the administrative details. Tricia Oberndorf added some comments regarding e-mail accounts. Sign-up with her or Carl Schmiedekamp if you need an account. If you have an account, please clean it up since they cost money to use, and cancel your account if it is not used. CDR Barbour then discussed the objectives of the meeting. The most immediate goal of the OSSWG is to work towards completion of the Delta Document. Other objectives were to redefine the subgroup structure, define the boundaries of the NGCR standardization areas, and develop balloting/commenting strategies for SAFENET I and POSIX.

Rich Bergman initiated the Subgroup reports with the Requirements Subgroup. The subgroup had concentrated on finalizing the Operational Concept Document (OCD). The OCD is now relatively stable. Their plans for the week were to fine tune the OCD and support the development of the Delta Document. Rich noted that requested changes to the Requirements in the OCD will be reviewed, but will be made at a later date.

The Available Technology Subgroup report was given by Karen Gordon for Jim Oblinger. The subgroup has been collecting data for the Delta Document. To facilitate this data collection, the 16 Service Classes were divided into 6 groups with assigned leaders as the central contact. The Available Technology Subgroup intended to concentrate on the Delta Document at this meeting.

Carl Schmiedekamp presented the Approach Subgroup status for Tom Conrad. Since the last meeting, a draft POA&M had been developed for discussion at this meeting. The Evaluation Process and Evaluation Results reports were in final publication, and the NIST POSIX Conformance Test Suite had been ordered. Plans for this meeting included revising the POA&M for Phase II of the OSSWG, discussing the OSSWG Reference Model, and finalizing the content of the Annual Report.

Following the OSSWG Subgroup reports, Tricia Oberndorf reviewed the results of the OSSWG's first POSIX meeting and the strategy of the OSSWG at POSIX. The Navy was generally well received at the Danvers, Massachusetts POSIX meeting. OSSWG members had volunteered for a large amount of work. As documents are completed, the balloting of them within the "dot" groups must be tracked. Each Star Group should review documents that pertain to them and share comments with the OSSWG. This is not to form

and vote as a block, but to share viewpoints across the OSSWG. The OSSWG must be careful not to overwhelm POSIX and look like an antagonistic group. The logistics of the balloting process should be discussed within the Star Groups as well as the means by which to advise the OSSWG.

The Star Group leaders reported on their status and findings from the Danvers meeting. CDR Barbour presented the Guide (1003.0) groups report. The purpose of the 1003.0 document is to provide guidance and consistency among other POSIX dot groups. A mock ballot (internal review) is scheduled for January 1991. It had been determined that POSIX should be used as an adjective. The 1003.0 group discussed what it means to be POSIX compliant. One idea is that to allow vendors to call their products POSIX, the products must be conformance tested to prove conformance to the standard. Conformance testing is important, especially when testing areas where the standards wording allows different interpretations. The 1003.0 group also struggled with the question of what a POSIX profile should be. They are looking at the ISO standards as an example.

Regarding the conformance test issues, Neil Henderson suggested that the OSSWG get together with the NIST and NGCR Conformance Test people to make quicker progress and save the repetition of work. Tricia noted that the NIST Conformance Suite only deals with 1003.1; however, the FIPS PUB is a good source to reference when writing the Operating Systems Interface (OSIF) standard. The OSSWG is meeting with NIST to work together and solve some of the OSSWG's issues.

Dan Juttelstad reported for Gail Holmes on the Conformance Test (1003.3) Star Group. The 1003.3 group has determined that each POSIX dot group will write their own test assertions with guidance from the Conformance Test group on how to write test assertions. The 1003.3 group will write the test methods for 1003.1 (1003.3.1) and 1003.2 (1003.3.2). The earliest completion date for 1003.3.1 is April 1991, 1003.3.2 should be complete in mid 1992. Gail's slides are included in Enclosure (2).

The Realtime (1003.4, 1003.13) report was given by Frank Prindle. Little was done on 1003.4 since the document is in the balloting process. Some discussion did occur on problems with the file system and name space. The bulk of the activity at the Danvers meeting was done on 1003.4a (p-threads). The Star Group needs to review the p-threads and see where they fit into the OSSWG requirements. Frank noted that there is a general sentiment among the community that some problems and inconsistencies exist for certain realtime systems. Frank suggested that the OSSWG should go into the next meeting with a written proposal on how the OSSWG requirements map to the p-threads. The 1003.13 group is developing Application Environment Profiles (AEPs). The OSSWG will have considerable influence in this group.

Mars Gralia presented the status of the Ada (1003.5) Star Group. The 1003.5 document tells how an Ada programmer gets access to POSIX unique services. The document went out for formal ballot in September 1990. The 1003.5 group has contacted the ISO community and sent a draft for them to look at, but due to the language independence issue, they were not asked to review it for acceptance. The 1003.5 group now

needs to write the test assertions for the document. A Project Authorization Request (PAR) is being written for an Ada binding to the realtime world. An area that must be looked at is Security within the Ada binding.

The Security (1003.6) Star Group status was reported by Mark Karan. The 1003.6 document is scheduled for a mock ballot in late 1990 and for formal ballot in the Summer of 1991. The 1003.6 document provides security for the basic 1003.1 document and is fairly complete. There are a number of issues regarding Ada and Realtime security which must be looked at. A POSIX liaison group has been formed between 1003.6, 1003.7, and 1003.8 to look at security issues in a distributed environment. Mark noted that the Star Group had looked at the 1003.6 document (Draft 5) versus the OSSWG requirements and found that a majority of the requirements are satisfied by POSIX.

Dan Juttelstad delivered the Distribution (1003.8, 1003.12, 1237, 1238) Star Group report for Greg Bussiere. The 1003.8 document covers Transparent File Access and is scheduled for a mock ballot in September 1990. The 1003.12 group is working on a Protocol Independent Interface. The 1003.13 group is a reorganization of an earlier group and is working on Name Space and Directory Services. The 1237 and 1238 groups are related to POSIX and cover RPC and FTAM, respectively.

After the Star Group reports, CDR Barbour then presented the official titles and availability of the four evaluation documents created and published by the OSSWG. The *Recommendation Report For The Next Generation Computer Resources (NGCR) Operating Systems Interface Standard Baseline* and *After-Action Report For The Next Generation Computer Resources (NGCR) Operating Systems Interface Standard Baseline Selection Process* reports have been published as NUSC technical documents and are available through NTIS and DTIC. The *Evaluation Process Report For The Next Generation Computer Resources (NGCR) Operating Systems Interface Baseline Selection* and *Evaluation Results Report For The Next Generation Computer Resources (NGCR) Operating Systems Interface Baseline Selection* reports are being published as NSWC technical documents and should be available in September 1990.

The Operational Concepts Document (OCD) was discussed by Dan Juttelstad. The latest version of the document is version 0.2, dated 11 July 1990. Dan reviewed the document chapters and their status. Chapters 1 and 2 are solid, while Chapters 3 and 4 need some final comments. Appendix A is the Reference Model. Carl Schmiedekamp has the latest version and the Approach Subgroup will work on updating it at this meeting. Appendix B contains the Requirements and is expected to be updated after the Delta Document is complete. Dan noted that the OCD has been written generically and could use some editing to add a POSIX flavor to it.

CDR Barbour added that realtime and Navy specific issues must not be forgotten in the Mission section of the OCD. It is important to keep in mind the "hard" problems which NGCR and the OSSWG must also solve. The OSSWG charter is for all Navy systems.

Jim Oblinger then discussed the Delta Document. He presented a proposed outline

for the document. Jim then presented a Service Class vs. Dot Group matrix. A copy of this and all of Jim's slides are contained in Enclosure (2). This matrix shows which POSIX dot groups are expected to fulfill the service class requirements. Jim noted that the matrix is probably incomplete and needs to be updated based on the POSIX meeting. Leaders had been assigned to collect the data for the Delta Document.

The data that is required has been standardized on a form. Jim noted that complete cross-reference of requirements to POSIX documents is desired because full compatibility of requirements is probably not going to occur. Also, it is important to identify inconsistencies between the POSIX documents, especially for the MIL-STD and prototypes. For each Service Class, a chart of the criteria versus POSIX dots was created. The matrix is then filled in with numbers to indicate if the particular document being looked at satisfies the criterion. The numbers were proposed as follows:

1 -> satisfies the requirement
2 -> interface exists, but is inappropriate
3 -> considered, but decision made not to include
4 -> not considered

It was noted that the situation where POSIX extensions discussed a requirement (criterion) and excluded it because the requirement is not a good idea to standardize on must be considered. In this instance, the OSSWG must then reconsider the requirement and determine if the requirement is really valid or needed.

Jim then reviewed the Delta Document process. First the requirements are analyzed and scored against POSIX. The requirements not satisfied will be listed. The lack of coverage for each requirement will be analyzed, and then strategies to include the requirement within POSIX or a means to satisfy it through other alternatives will be developed. A question was raised at to whether the Delta Document will address POSIX capabilities not contained in the OSSWG Requirements. It was felt that this could not be done in the timeframe of the Delta Document since it will result in the documentation of all of POSIX.

A report on the NGCR Co-Chairs Meeting was given by CDR Barbour. He presented issues that were discussed at the Co-Chairs meeting and recommendations that were made to the Program Office System Engineer (Frank Deckelman). CDR Barbour also reported on the Realtime Working Group which is studying some cross working group issues that have been identified by the Co-Chairs. He noted that the realtime requirement is not necessarily deterministic performance, but "bounded" performance. The issues raised in the Co-Chairs meetings are designed to attain an NGCR viewpoint with participation from all affected working groups. The Co-Chairs meet on a quarterly basis.

The meeting broke into the Subgroup meetings for the remainder of the day.

Wednesday, 29 August.

The OSSWG Subgroups met for the morning and for half of the afternoon. The OSSWG then broke into the POSIX Star Groups to strategize for the remainder of the day.

Thursday, 30 August

The new OSSWG Subgroups met for the early part of the morning. The new subgroups are: Approach, OSIF Standard Evolution, and Exploration & Transition. The OSSWG then reconvened for the remainder of the meeting. CDR Barbour discussed the new subgroups. He noted that, based on the sign-up, there was a disproportionate distribution of people in the three subgroups. The most important item for the OSSWG to concentrate on is the Delta Document which falls under the OSIF Standard Evolution Subgroup. Everyone, regardless of the subgroup they are in, should help with the writing of the Delta Document.

The Star Groups were given an opportunity to relate any important discussions that occurred in their meetings. Mars Gralia discussed a balloting issue that arose in the Ada Star Group (1003.5). He noted that the OSSWG should not block vote on the 1003.5 ballot. However, ideas on different positions should be shared among the OSSWG via e-mail. CDR Barbour added that e-mail is the fastest and best way to communicate and distribute information to the OSSWG. He suggested the use of Booz, Allen to help with the location and dissemination of documents if needed. Contact Mark Karan (301-951-2739) for information. It was suggested that the OSSWG expand its e-mail on POSIX issues to the IEEE POSIX e-mail group.

A discussion took place on the number of people in each Star Group and attending the POSIX dot groups. It was felt that some areas were well covered and some needed more people. The suggestion was made to put out a list of the current coverage in the POSIX dot groups and the number of people suggested to be in the group.

The old and new OSSWG Subgroups then reported on their progress, plans, and issues. Rich Bergman started with the final Requirements Subgroup report. The OCD had been wrapped-up. Comments will still be handled by Dan Juttelstad.

Jim Oblinger provided the final Available Technology report and the OSIF Standards Evolution Report since the majority of the work was rolled over to the new subgroup. The group discussed the Delta Document and analyzed the data that had already been received. Jim presented one of the data sheets that had been completed in the subgroup meeting. The Delta Document Outline had been modified to include a look at the Big 6 requirements and how they are satisfied. The 6 groups that were organized to collect the data will be maintained to ensure the completion of the requirements data collection. Of the 156 requirements to review, data had been received on 116 (75%) of them. 38 of the requirements were discussed in the subgroup which led to the writing of some of the analysis sections. Jim then discussed the plans to complete the Delta Document by the end of the year.

It is important to identify which documents are part of the OSIF baseline for the

Delta Document. This will be based on which documents were used for the data analysis. The latest versions of each of the documents must also be maintained. Carl Schmiedekamp will add a star group directory and a POSIX dot group directory in the OSSWG Archives. Jim also mentioned that the Available Technology Report is going through the process to become a Technical Document and will be published soon.

The Approach Subgroup report, both old and new, was presented by Tom Conrad. The new POA&M has been revised and refined. The subgroup discussed the Reference Model status and have a plan to update it by December. The Annual Report was also discussed. The presentation slides are included in Enclosure (2).

Tom presented the new OSSWG organization and the primary relationships between groups. The OSSWG consists of the new subgroups previously stated as well as the POSIX Star Groups. Each OSSWG member belongs to two groups: one of the subgroups and one of the Star Groups. External to the OSSWG, the subgroups communicate with the other NGCR Working Groups and the Star Groups communicate with the POSIX dot groups.

Tom briefly discussed the dot groups in POSIX which are not currently being attended by the OSSWG but probably should be. The dot groups identified were: 1003.1, 1003.2, 1003.7, and 1003.14).

A timeline of the expected products from the subgroups was presented. The subgroup responsibilities were explained and the subgroup membership was discussed.

Next, the Star Group membership and responsibilities were discussed. The responsibilities include POSIX proposal packages, ballot identification, Ada bindings, and meeting reports. The Star Group Meeting Reports are to be one to two page reports providing information on meetings, ballots, and key issues from the POSIX dot group meetings. A template for the reports is included in Enclosure (2). The reports should be completed within two weeks of the meeting and should be output on e-mail to the Executive Committee (execcomm@tecr.nosc.mil). Star Group Meeting Reports should be written for the Danvers POSIX meeting by the appropriate individual.

Tom then discussed the Annual Report and presented a high-level outline. The report will consist of three volumes: Executive Summary, Record of Progress, and Principal Products. This report will be compiled in October.

The meeting wrap-up began with Tricia Oberndorf. The Navy Labs should note that the four Technical Documents produced are not being sent directly to the OSSWG participant's iab supervisors. It would be helpful if each lab made copies of the documents and forwarded them to their Lab Commander and Technical Director (TD) with a personal note indicating the current efforts and accomplishments of the OSSWG and NGCR.

Regarding e-mail, contact Carl Schmiedekamp if you need an account. Tricia asked if the group felt that new e-mail lists were needed for the Star Groups. Contact Tricia with the names of the group to set up a list. For balloting issues, messages and opinions should

be sent to the entire OSSWG (osswg@tecr.nosc.mil). If you want to be added to an e-mail list, send a message to Tricia (tricia@nadc.navy.mil).

Tricia noted that the Ada 9X effort may need to be followed by the OSSWG. Ada 9X is updating the Ada Language Reference Manual. They are targeting a 1993 completion date for a new standard which should be backward compatible. It was suggested that a briefing to the OSSWG on the 9X effort be arranged. Del Swanson will investigate this.

CDR Barbour then provided some final comments. He noted the next meeting which will be held in conjunction with POSIX. The POSIX meeting will be held at the Westin Hotel in Seattle, Washington from 15-19 October 1990. The OSSWG meetings will be held on the 16th and 18th from 1800 to 2100 hours.

The meeting objectives were reviewed and the progress discussed. The OCD is almost complete, the progress made on the Delta Document was better than expected, the new subgroups have been set, and the standards boundaries were discussed. CDR Barbour also noted that a number of comments had been received to establish balloting procedures and strategies.

CDR Barbour discussed the current Joint Service drive for NGCR. It is starting to happen at the upper levels, based on a letter from an Assistant Secretary of Defense. CDR Barbour requested that everyone keep this in mind and contact him if you have any input. Talking and promoting NGCR will help give the NGCR effort visibility in the community and keep it moving forward. The meeting was then adjourned.

Part 3

Principal Products

The major products produced by OSSWG from January 1989 through August 1990 include:

(1) White Paper on Network Operating Systems Standards (8/88)
(2) Plan of Action and Milestones for the OSSWG
(3) DID for Operational Concept Document for NGCR Operating System Standard (8/89)
(4) NGCR OSSWG Reference Model, Version 1.02 (8/89)
(5) NGCR OSSWG Available Technology Report Version 1.3 (9/90)
(6) Operating System Interface Standard Requirements, Version 2.0 (12/89)
(7)* Evaluation Process Report for NGCR Operating Systems Interface Baseline Selection (5/90)
(8)* Evaluation Results Report for NGCR Operating Systems Interface Baseline Selection (5/90)
(9)* Recommendation Report for the NGCR Operating System Interface Standard Baseline (6/90)
(10)* After-Action Report for the NGCR Operating System Interface Standard Baseline Selection Process (6/90)

* These documents are not included in entirety but appended by reference.

NOSC White Paper

on

Network Operating Systems Standards

August, 1988

1 Introduction

The Next Generation Computer Resources (NGCR) Program is to provide the standardization of Navy mission critical computer interfaces and computer component interfaces. With these standardized interfaces, industry will be better able to provide computing resources that meet Navy needs.

The interface standards are to be widely available (i.e. non-proprietary) and, if possible, widely utilized within industry.

The Network Operating System Interface Standard (NOSIS), the subject of this paper, is one of the sets of standards which is essential to the timely and cost effective acquisition of the majority of the next generation of Navy mission critical computing systems. NOSIS assists the Navy in efficiently providing a wide range of performance, compatible computing services, and functionality levels.

2 Scope

The NGCR interface standards, while being incrementally developed, are to be sufficiently in place so that the Navy can begin acquiring systems utilizing those standards by 1994. Prototype systems using the NOSI Standards are to be developed, with a contract award scheduled for June 1990.

The period of NOS standards development begins in FY89 and continues through FY95 and beyond. The initial NOS standards will be available for use in acquisitions starting in FY93.

The initial range of applications include as many types of computing as possible from just above the single dedicated processor to as high as can be obtained on networked, heterogeneous, modularized backplane bus architecture computing systems. Networking to be done using NGCR LAN standards and, as appropriate, other MIL-STD links.

3 Issues

3.1 Technical

There are several areas of technical concern which should be considered during the development of a set of specifications for a family of real-time distributed target operating systems. Some of the major areas of concern are listed below with a brief description. They are considered essential characteristics of the operating system. A level of transparency offered to the user by the operating system is assumed. No current distributed operating system design adequately addresses the requirements, especially in the areas of real-time constraints, multi-level security and fault tolmiolerance. The following list in not in any prioritized order.

Real-Time

A tactical real-time command and control system must be able to meet the timing requirements of a variety of periodic and aperiodic requests. Research has shown that speed alone does not adequately solve the problem of meeting these requirements. The mechanisms for supporting real-time scheduling of system resources requires the integration of the hardware ahe operating system subcomponents in a deterministic and predictable manner.

Distribution / Networking

The C3 arena is naturally distributed and complex; therefore future systems must integrate distributed resources. Not only must the target operating system provide communication mechanisms, but it must use them in such a way as to unify this distributed set of system resources. All this must be done while still considering the real-time requirements of the system.

Heterogeneity of Functions / Processing Elements

The target operating system must support a heterogeneous environment to allow for the incorporation of new technology and new mission requirements. This supports one of the objectives which is to avoid a dependency on proprietary products. Heterogeneity should be supported at many levels. Support for standard programming languages such as Ada increases program portability. Possibly inconsistent object formats is a problem that needs to be considered. The ability to convert data representations between a variety of targets is an important operating system function to support. There are other areas such as the file system structure and symbolic naming that need to be considered in terms of a heterogeneous environment.

Array / Parallel Processing

The target operating system must be able to support various

computer architectures including parallel or multiprocessors. The operating system should be able to take advantage of such systems and at the same time it must work in cooperation with the rest of the distributed system.

Recovery / Damage Control/ Fault Tolerance / Survivability

Survivability and reliability are extremely important in a command and control environment. It is unacceptable to experience a total loss of functionality and availability due to a single system failure. The target operating system must offer a fault tolerant environment through the dynamic replication and duplication of services and resources. This environment must be fault tolerant at all levels from the network to the application and work as a functional whole with the rest of the system.

Security

The target operating system should be able to protect system integrity from inadvertent or malicious misuse. The system should allow for multiple concurrent levels of security within a node as well as across the distributed system. The security mechanism should conform to available and evolving DoD standards as appropriate. Security is a particularly difficult issue to solve when coupled with the performance requirements of tactical C2 systems. Speed requirements along could push security mechanism into the hardware.

Data Flow and Throughput

Navy systems and applications are typically data and cpu intensive. It is important that the target operating system incorporate mechanisms for a variety of data flow requirements and that the functions meet certain levels of performance requirements. It is equally as important and difficult to determine what are the performance requirements of C2 applications. Dataflow and throughput must be consider all levels including the system and network.

Performance

Database

I/O

Family / Architecture

Interface

user, applications, backplane, HW/network, Lan) application - language binding to application, user - cli definition, protocol vs procedure calls. OS needs to interface with a bunch of thinks. System services will probably be queued separately from application. resource management issues. I/O, backplane, hardware, network.

Host / Target Interface - relationship

host is for example a vax. It is where you do the development, programming support, debugging and testing. Explain integration of the two. Do you physically carry over a tape, have a direction connection, or are connected by satellite to the ship (new software can be load on target machine over the satellite?). Issues is how to get application to the target machine from the host machine or get status reports from the target machine during run time. Seems to be similar to D2 to D1 hand off or at least the other direction.

Domain 2 to Domain 1 Hand-Off

In addition to the essential characteristics, it is also important to consider the critical components necessary to support these characteristics. Some of the major services considered necessary are system-wide resource management, communications, timing services, synchronization, naming, addressing, access control, authentication and storage management. Research supports the use of objects, invocations, and threads in the development of distributed systems. There is need to offer a variety of real-time scheduling mechanisms and incorporate flexibility and evolvability through the use of policy modules.

3.2 Policy

4 Approach

The primary objective of the NOSISWG will be the development of a set of interface standards for a family of real-time distributed target operating systems. In support of this objective, it will also be necessary to develop one or more prototype implementations and to generate a variety of accompanying documents, including at least the following:

operational concept

requirements (with rationale)

rationale for the set of interface standards

user and implementer guides

The NOSISWG should have primary responsibility for all decisions made with respect to the operating system interface specification and accompanying products. It should be structured analogously to the existing NGCR working groups, with a government Chairman and Co-Chairman and a mixture of government, university and industry participants. Meetings should be at least quarterly, possibly supplemented by more frequent meetings of individual subgroups.

Before the NOSISWG is first convened, a lead laboratory should be chosen and tasked to do further planning. This planning should further develop and elaborate on the suggestions presented here for organization, issues and products. The first NOSISWG meeting should be attended by only government personnel. This is to ensure coherence and direction of the government objectives and requirements prior to exposure of these to the general community. Such an initial government meeting can be pursued in parallel with the solicitation of initial information from industry and universities.

Government participants should be solicited from at least each of the Navy laboratories and PDSS activities. Other sources of relevant expertise should also be investigated and tapped if possible, including Navy testing activities, development and PDSS organizations from the other services, and other federal agencies, such as DARPA and NASA.

Industry and university participants should be solicited both from known sources and through open solicitations such as in the CBD. It should be assumed both that the government does not have sufficient qualified personnel by itself to successfully complete this project and that volunteers (whether from government, university or industry) cannot be expected to be sufficiently regular or dependable. Thus plans should be made to have two kinds of support contracts. One would be administrative/secretarial in nature, the other technical. The technical "contract" could in fact be several contracts, each for a different sort of expertise, or it could be one contract awarded to a sufficiently diverse team.

One of the first activities of the NOSISWG should be the formulation of a charter. This activity will serve to focus and channel the thinking of the participants. Any subgroups should also formulate charters for their special objectives.

The NOSISWG should be free to form subgroup structures as they are needed. These will most likely respond to different needs at different stages in the life of the NOSIS activity. Initially it is suggested that a subgroup structure be formed which is oriented around the different kinds of issues presented in the last section. These issues can be grouped in ways which afford an opportunity for participants with similar interests and backgrounds to discuss a logical group of related issues to better describe them and to get a better understanding of their role in the entire NOSIS effort. Later it is likely that a subgroup structure oriented around the products or around a set of orthogonal concerns would be more productive. One

such structure might have a subgroup for each of Requirements, Scope (i.e., the family and other architectural issues), Availability (of technology to meet the emerging requirements) and Policy (such as the issues involved in standardization).

The last item above is an important one for the NOSISWG to keep in mind. The generation of a standard involves issues, such as those discussed in the previous section, that are far removed from the technology involved in answering the requirements. These must be taken into account at every stage along the way.

5 Available Technology

No operating system exists which adequately meets the requirements of tactical C2 systems. However, there exists a large amount of expertise in government, universities and industry. The level of work being done by these various groups ranges from purely theoretical to attempts to produce a product. The following is meant to highlight some of the more extensive work being done and is by no means to be considered a complete list. These groups could potentially provide value input to the development of NOSIS.

5.1 Technical Groups

Naval Ocean Systems Center (NOSC)

NOSC is developing distributed operating systems (DOS) for Navy C2 systems, under funding from ONT. The major concerns of the NOSC group involve Navy distributed C2 system requirements and their relationship to domain 1 and domain 2 operating system issues. The approach is to monitor, participate, and influence DOS research and development with respect to these Navy requirements. This involvement has included work with RADC, NSWC, NUSC, CECOM, JDL, DARPA, CMU, UVA, BBN and IBM as well as numerous others. Experience has been gained through the development of a DOS testbed which currently uses Cronus and will include Mach and ARTS in FY89. Point of contact: Les Anderson, Code 443, NOSC, San Diego, CA, 92152-5000, (619) 553-4139.

Rome Air Development Center (RADC)

RADC is developing a distributed operating system to meet the requirements of BM/C3 systems. They fund numerous distributed operating system projects such as those at BBN, CMU, KSR, and Honeywell which has lead to the development of systems such as Cronus and Alpha. They are presently participating in a tri-service DOS experiment with NOSC and CECOM funded by the Joint Directors of Laboratories (JDL). Point of contact: Dick Metzger, RADC, Griffiss Air Force Base, NY, 13441-5700, (315) 330-2066.

Naval Surface Weapons Center (NSWC)

NSWC is the primary support lab for the AEGIS tactical real-time system. They (group N35) have extensive knowledge of current real-time Navy executives and the needs of real-time applications. In the past, they have funded real-time distributed operating systems research at CMU and are presently working with NOSC on issues dealing with Navy tactical computers on LAN's. Point of contact: Daniel Green, Bob Harrison, NSWC, Dahlgren, VA, 22448, (703) 663-4585.

CECOM

In the past, CECOM has done work in the area of distributed operating systems. They funded the Command and Control Information Utility (CCIU), an experimental distributed information processing system developed by JPL. They are presently participating in a tri-service DOS experiment with NOSC and RADC funded by JDL. Point of contact: Frank Holloran, CECOM, Fort Monmouth, NY, (201) 554-4158.

Naval Underwater Systems Center (NUSC)

One of the groups at NUSC is working in the area of distributed systems and distributed operating systems. They published a book entitled "Design of Distributed Operating Systems" and several other papers on fault tolerant and real-time systems. Point of contact: Paul J. Fortier, NUSC, Newport Laboratory, Code 2222, Newport, RI, 02840, (401) 841-3703.

Ada Runtime Environment Working Group (ARTEWG)

The Ada Runtime Environment Working Group (ARTEWG) is a working group sponsored by the Association for Computing Machinery (ACM) Special Interest Group on Ada (SIGAda). Its goals are to establish conventions, criteria, and guidelines for Ada runtime environments that facilitate the reusability and transportability of Ada program components, improve the performance of those components, and provide a framework which can be used to evaluate Ada runtime systems. Since Ada runtime will be a critical part of any NGCR system, it is very important that this work be taken into account. In addition, the ARTEWG represents a valuable group of volunteers who have put a lot of time and effort into understanding the runtime needs of real-time systems and who are very knowledgeable about many of the issues facing the NOSISWG. Point of contact: Mike Kamrad, Honeywell Systems & Research Center, Minneapolis, MN, (612) 782-7321.

National Aviation and Space Agency (NASA)

The National Aeronautics and Space Administration (NASA) Space Station project is intended to field an elaborate space station facility in the 1990's. The system will be highly computer-dependent and involves many of the key features of the NOSIS: real-time, distributed, heterogeneous, etc. This group of people has been gathering information and experience for the last few years and would be a valuable source of insight into potential NOSIS issues and

challenges. Point of contact: Ed Chevers, NASA Johnson Space Center, Houston, TX, (713) 483-4281.

Advance Real-Time Technology (ART)

ART is an open project interested in the development of advanced real-time technologies. The project is funded by ONR and presently includes efforts by CMU, SEI and IBM-FSD. The objective of the ART project is to develop theoretical foundations, distributed real-time system technology and programming language support that will facilitate the development of distributed real-time systems with understandable, predictable, and maintainable behavior. They are interested in the development of real-time scheduling theory, distributed real-time operating systems and distributed real-time databases. The project is presently developing a real-time DOS called ARTS and an experimental system called Real-Time Mach. Point of contact: Hide Tokuda, Computer Science Department, Carnegie-Mellon University, Pittsburgh, PA, 15213, (412) 268-7672.

BBN

BBN is involved in the development of network and network environments. They recently ported the Mach network operating system to their Butterfly multiprocessor. They are also responsible for developing the Cronus distributed operating system. See Cronus in section 5.3 for more details. Point of contact: Andres Echenique, BBN, 10 Moulton Street, Cambridge, MA, 02238, (617) 873-4304.

Carnegie-Mellon University (CMU)

There are several groups at CMU involved in research in the areas of distributed operating systems and real-time distributed operating systems. See ART, ARTS, Alpha, Mach in sections 5.1 and 5.2 for more details.

IBM

The IBM research groups at Owego and Manassas have been heavily involved in real-time scheduling issues. The IBM-FSD group is currently involved with the ART project and has developed a real-time survivable network prototype. Points of contact: C. Douglass Locke, IBM, Route 17C, Owego, NY, 13827, (607) 751-4291. Pat Watson, IBM-FSD, 9500 Godwin Drive, Manassas, VA, 22110, (703) 367-4536.

Kendall Square Research (KSR)

KSR is using the technology developed at CMU to produce a commercial version of the Alpha real-time distributed operating system for their new multiprocessor to be released in mid FY89. Point of contact: E. Douglas Jensen, Kendall Square Research, One Kendal Square, Cambridge, MA, 02139, (617) 494-1146.

Software Engineering Institute (SEI)

SEI is currently involved with the evaluation of the Ada language with respect to its ability to support the specifications of real-time system scheduling. They are working with VERDIX, Honeywell, IBM-FSD and CMU on a variety of real-time distributed system issues. They are presently developing a real-time Ada operating system kernel capable of running in a distributed processing environment. Point of contact: Lui Sha, SEI, Pittsburgh, PA, 15213, (412) 268-7868.

Other

There is other research and development in the areas of real-time and distributed systems. All of these groups, especially the ones mentioned above, have value insight in to real-time and distributed systems. IBM-FSD is participating in the ART project funded by ONR under the Real-Time Initiative Program. There is an operating systems group at the Defense Systems Branch of UNISYS in Camarillo doing work with Mach and other prototype systems. Honeywell has been researching issues involved with real-time Ada and have worked closely with the ART group at SEI. Points of contact available.

5.2 Real-Time Distributed Operating Systems

ARTS

ARTS is the real-time distributed operating system being developed by CMU and the ART group. See ART in section 5.1 for more details. Point of contact: Hide Tokuda, Computer Science Department, Carnegie-Mellon University, Pittsburgh, PA, 15213, (412) 268-7672.

Alpha

The Alpha kernel is being developed at CMU with funding from ONR, RADC and NOSC. It is intended to support a range of system solutions that effectively meet the requirements of various reliable, distributed real-time command and control applications. It supports decentralized management of global resources and has kernel level support for atomic transactions and replication. It is presently being used as the base operating system for a new multiprocessor being developed by Kendall Square Research and was used for an SDI demonstration by General Dynamics in FY88. Point of contact: J. Duane Northcutt, Computer Science Department, Carnegie-Mellon University, Pittsburgh, PA, 15213, (412) 268-7678.

5.3 Distributed Operating Systems

Cronus

Cronus is a distributed operating system being developed by BBN and is funded by RADC, NOSC and ESD. It incorporates many desirable DOS features such as heterogeneity, transparency and object oriented programming as well as high level features such as survivability and replication mechanisms, multi-cluster and database access and distributed monitoring and control facilities. Cronus is presently being used by several Navy projects such as Fleet Command and Control Battle Management Program (FCCBMP). It is also being used as a basis for study at NOSC of Navy DOS requirements. Point of contact: Andres Echenique, BBN, 10 Moulton Street, Cambridge, MA, 02238, (617) 873-4304.

Mach

Mach is a multiprocessor-oriented operating system for a distributed environment being developed at CMU and is funded by DARPA. It approaches issues involved with multiprocessors, heterogeneity, transparency, and object oriented programming. Mach is presently being used by several projects at CMU and being used and extended by a number of corporations, universities and research laboratories. NOSC has plans to add Mach to its DOS testbed in FY88-89. Point of contact: Rich Rashid, Computer Science Department, Carnegie-Mellon University, Pittsburgh, PA, 15213, (412) 268-2627.

Sprite

Sprite is an experimental network operating system under development at the University of California at Berkeley and is funded by DARPA. Motivation for the new operating system came from trends toward networks, large memories and multiprocessors. Sprite is part of a large project called SPUR whose goal is to develop a high-performance multiprocessor workstation with special hardware support for LISP. Sprite focuses on issues involved with transparent network file systems, large variable-size file caches, shared address spaces, and process migration in a distributed environment. Point of contact: John Ousterhout, Computer Science Division, University of California, Berkeley, CA, 94720, (415) 642-0865.

V Distributed System

The V Distributed System is a network operating system being developed at Stanford University and is funded by DARPA, the National Science Foundation and AT&T Information Systems. It is used to explore issues in distributed systems and focuses on areas such as high performance interprocess communication including multicast, process migration and the distributed scheduling of programs. There is also research being conducted in the areas of replication, distributed atomic transaction management and multiprocessors. Point of contact: David Cheriton, Stanford University, Computer Science

Department, Stanford, CA, 94305-2140, (415) 723-1054.

Other

There is other real-time and distributed operating system work being conducted at universities such as the University of Massachusetts, Georgia Institute of Technology, and the University of Virginia. The list is too extensive to be completely list in this paper.

5.3.1 Real-Time Centralized Operating Systems

ALS/N

ALS/N (Ada Language System / Navy) is the Ada compiler being developed for the UYK43. The compiler will be test at NOSC in FY 89.

Harmony

Harmony originally came out of the Canadian National Research Council and is now developed and marketed by DY-4 Systems (Nepean, Ontario). Harmony is a real-time operating system that was originally designed to be a multiprocessing kernel. A copy of the kernel resides in each processor. Although other kernels such as MTOS and VRTX use the same arrangement, Harmony data structures aren't replicated. The kernel for each processor has its own data structures, and there's typically little sharing of data between processors. Since global memory is used less, contention for the multiprocessing bus - a problem that plagues many multiprocessing kernels - is reduced.

Standard Distributed Executive (SDEX)

There are two versions of SDEX, one for the UYK-7 and the other developed by the Canadian government for the Canadian frigate program, targeted for the UYK-44. The SDEX has real-time priority scheduling capabilities as well as message handling capabilities. It is currently being used in a variety of mission critical systems in the Navy.

VRTX

VRTX is manufactured by Ready Systems (Palo Alto, CA). Chief among VRTX's facilities is multitasking. The task is to real-time programming what the procedure is to more conventional programming; a structural unit that can be considered separately from other units. VRTX has been designed to maximize its performance for high-priority tasks, unlike non-real-time systems, which are designed to be "fair" and allocate an equal overhead to all tasks. VRTX is also a compact system, occupying less than 6K bytes of code space. Most importantly, VRTX is deterministic. Its behavior can be completely predicted in all circumstances.

VxWorks

VxWorks operating system from Wind River Systems (Emeryville, CA) is designed to debug and run real-time tasks developed on a Unix system. Target computers can use 68000, 68010, or 68020 processors, and can run alone or networked with other computers that run VxWorks or Berkeley 4.2 Unix over Ethernet or a backplane bus. VxWorks uses calls very similar to Unix calls.

5.4 Interface Standards

CAIS

The Common Ada Programming Support Environment (APSE) Interface Set (CAIS) (DOD-STD-1838) is a set of Kernel APSE (KAPSE) level interfaces designed to provide a portability base for tools written in Ada. It is in the form of Ada packages. The CAIS provides services for an object management system, process management and input/output. As an interface set for the host (APSE) system, it is not a candidate itself for the NOSIS. But many of its features and the experience gained in its development may well be relevant to the NOSISWG effort. Point of contact: Duston Hayward, Code 423, NOSC, San Diego, CA, 92152-5000, (619) 553-4067.

Microprocessor Operating Systems Interfaces (MOSI)

The Microprocessor Operating Systems Interfaces specification is an Institute of Electrical and Electronics Engineers (IEEE) trial-use standard issued in December 1985. It applies to program interfaces for microprocessor operating systems; the interfaces are used by microprocessor applications to interface with operating system services. It is pointed out in the literature that the standard is not an operating system standard; it defines a program interface to any operating system. Its stated objective is to facilitate the writing of portable application programs or sys. Like the CAIS, its relevance to NOSIS is most likely to be in features that can be studied and possibly applied to the NOSIS. Point of contact: Secretary, IEEE Standards Board, 345 East 47th Street, New York, NY 10017.

Portable Operating System for Computer Environments (POSIX)

The UNIX operating system has its roots in Multics, a multi-user, multi-tasking operating system developed in the late 1960s at MIT. UNIX was created as a software development system by Bell Laboratories in 1969, and underwent a number of revisions by Bell, and was modified and enhanced by others. UNIX System III has served as the basis for many of the proprietary UNIX derived operating systems available today. The new UNIX System V is the latest Bell revision. Capabilities not present in Bell's latest system version are available from other sources, most notably the University of California at Berkeley's computing center. The latest version of

Berkeley UNIX is 4.3.

IEEE Standard 1003, IEEE Standard Portable Operating System for Computer Environments is an attempt to define a standard operating system interface and environment based on the UNIX Operating System. They are to develop documentation to support application portability at the source level. This is intended for systems implementors and applications software implementors. There are several subgroups within IEEE Standard 1003 considering issues such as security, real-time, verification and Ada interface.

Open System Interface (OSI)

OSI is a set of standards being developed by the International Standards Organization to enable heterogeneous computer systems to interconnect and interwork regardless of their manufacturer's models, complexity, or age. There is a seven-layer structure called the OSI Reference Model and it provides the framework for defining the requirements and standards. Many of the layers have been standardized and should be considered by NOSISWG whenever appropriate. It is unclear how an operating system properly fits into the OSI model.

5.5 OS Areas of Consideration

Define:

distributed:

real-time:

network:

PLAN OF ACTION AND MILESTONES
FOR THE
OPERATING SYSTEMS STANDARDS WORKING GROUP (OSSWG)

Background

The Navy has a long history of developing and using standard computer ducts. When computer technology was in its infancy, the Navy wielded nificant influence in the market, setting its own requirements and eloping its own computer designs, including Instruction Set Architectures SAs). Standard computer implementations (i.e., buying "boxes") and upward patible ISAs have been the foundation of the Navy's computer policy. This icy has been motivated by the fact that software can adapt a common puter design to meet many different applications. This allowed one design achieve large production runs and lowered production, operation, and ntenance costs. By limiting the number of computer types, it has been sible to develop and maintain computer programs for many applications with set of Navy-owned and -maintained support software for each computer e. Fleet spares, training, documentation, and support equipment uirements have been minimized, thereby increasing operational ilability.

The Fleet today remains, as it always has been, highly dependent on its lity to support the weapon systems which provide the force structure defense. Uniqueness and lack of interchangeability are two of the st enemies we face, because they each require more and more elaborate rastructures to maintain weapon systems readiness. Nevertheless, one not ignore the smaller role the Navy plays in the development of puter systems compared to several years ago. Nor can the desire to lize Commercial-Off-The-Shelf (COTS) and Non-Development Item (NDI) tems be ignored.

The advance of semiconductor technology has been extreme in the recent t, and shows no sign of decline. The challenge, therefore, is how insert new technology based on the tremendous commercial developments, hout sacrificing the operational requirements of the Fleet. In the cess, we must not allow technology to be used simply for the sake of hnology: the weapon system requirements must be well understood and t firmly in grasp.

No method currently exists to insert new technology into the Fleet at breakneck pace at which it is being introduced. However, through the of innovative development processes such as:

(a) extensive front-end work with industry to define standards,
(b) heavy utilization of simulation tools for design and test, and
(c) an overall plan for continuity for new platform starts and upgrades,

will be able to meet both Fleet requirements and Congressional desires. these assumptions, the NGCR Program is based.

The NGCR Program's approach is an open systems architecture based on the ablishment of standards in 10 interface areas:

Multisystem Interconnects:
- Local Area Network - SAFENET I
- Local Area Network - SAFENET II
- High Performance Local Area Network

Multiprocessor Interconnects:
- Initial Backplane
- High Performance Backplane
- Switch Network

Operating System

Data Base Management System
Programming Support Environment
Graphics Language/Interface

Application of these interface standards will change the Navy's approach from one of buying standard computers to one of procuring computing resources which satisfy the interfaces defined by the standards. These standards will be applied at the project level rather than a Navy-wide procurement level.

These interface standards will be based, to the greatest extent possible, on existing industry standards. In cases where existing industry standards do not meet Navy mission-critical needs, the approach is to further enhance the existing standards jointly with industry, thus assuring the most widely-accepted set of commercially-based interface standards possible.

The Operational Requirements describe some of the desired characteristics of the computer systems which can be procured using the new interface standards:

- a full-range family of computing resources, related through a set of interface standards, in a wide range of performance levels; software compatibility at appropriate levels is a necessary part of the "family" relationship

- integration of multiple, dissimilar (heterogeneous) processors

- internal and external standard interconnection; i.e., an internal computer interconnection (bus) to provide for growth in internal capability by configuring more modules and an external interface to provide for combining computing systems

- incremental computing system growth; i.e., if a new function is needed, new modules or computers would be added to a system, and adding the new components would not require replacing the old system: the new components would perform the new function in cooperation with the old; the simplest realization of this is interconnection of computing elements on expandable backplane busses and local area networks; the ultimate realization of this is automatic, global, dynamic task allocation among computing elements using a variety of interconnections (this requires a high degree of software transportability).

An operating system interface standard is a key element in the success of NGCR. The function of the OS is to control operation of all the computing system hardware and software elements in a coordinated, uniform manner that is consistent with the needs of real-time applications. The OS capabilities include system initialization and fault tolerance/recovery, global resource allocation, and interprocess communication. The OS will have components in each processing element. The OS interface standard will not be a design of the OS component of each processing system but will, in part, be a specification of an application task interface common to all computing elements. (The appropriate specification level for this interface must be determined.) This will provide a basis for system-wide dynamic task and resource allocation. Global dynamic task and resource allocation is the basis for system-wide fault tolerance and recovery in heterogeneous processing systems. The OS will provide the ability to achieve multi-level security at the system level. Conformance to other Navy directives requires that the OS be Ada-oriented.

The approach to the operating system has not yet been determined, but it is clear that the above system characteristics and OS requirements have some implications for the OS interface standard. Most important of these is that the OS "standard" will most likely be a family of compatible standards, although the exact nature of the family relationship is yet to be determined.

Objective

The primary objective of the OSSWG is to establish a commer-
.ly-based family of operating systems interface standards for
in the development and deployment of Navy MCCR applications
:ems in the mid-1990s and beyond. In order to accomplish this
ective, the OSSWG shall: (a) identify Navy applications systems
iirements; (b) identify existing and evolving operating systems
nology; (c) determine the applicability of identified opera-
systems capabilities to the requirements; and (d) make re-
endations to the Program Office (SPAWAR-324), particularly
arding the establishment of NGCR operating system interface
dards.

Organization

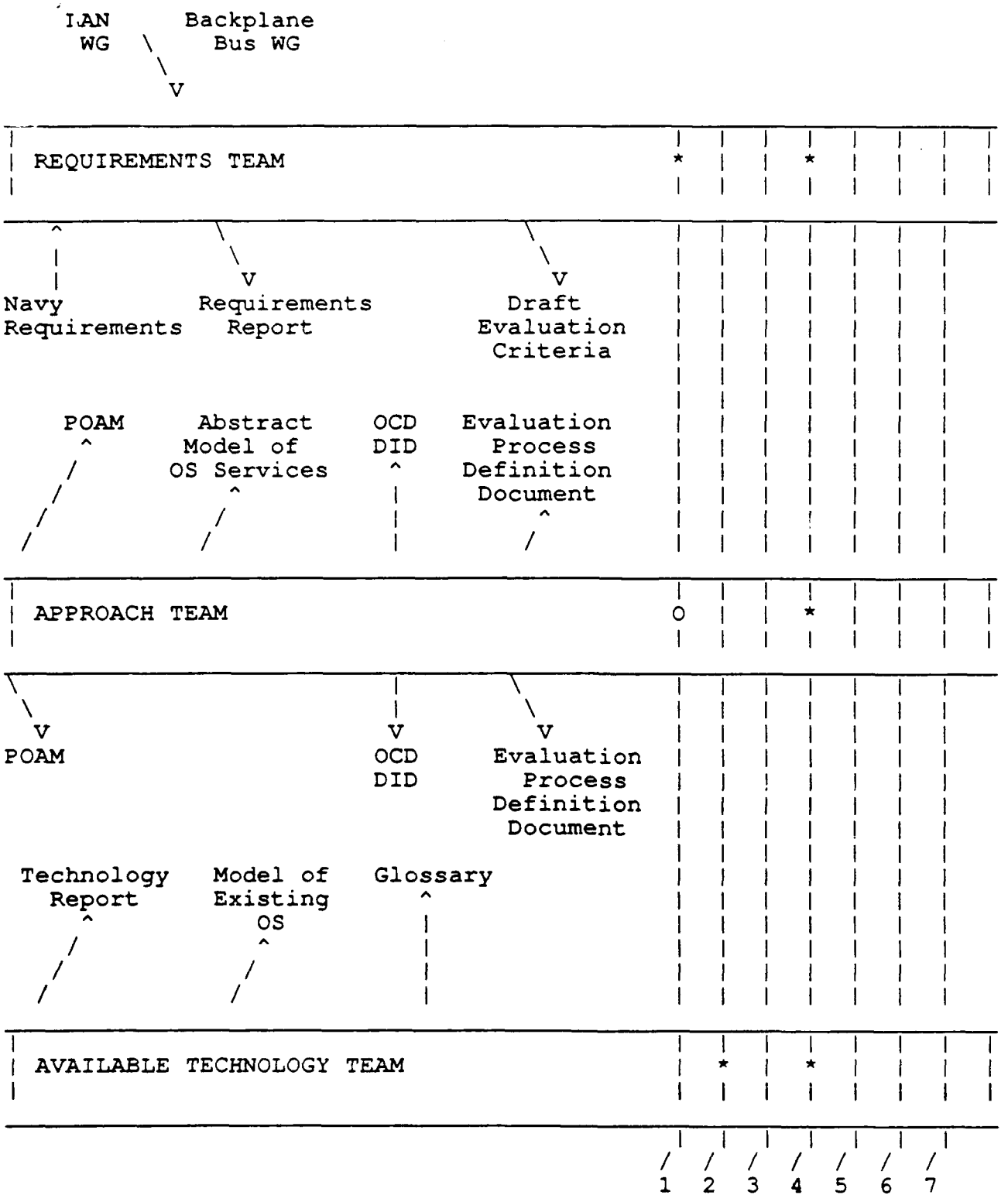
The OSSWG will be structured analagously to the existing NGCR
ting groups, with a Navy military Chairman, a Navy civilian Co-
rman and a mixture of government, industry, and university
icipants. The OSSWG will form subgroups as they are needed.
e will respond to different needs at different stages in the
se of the OSSWG activity. All subgroups will be chaired by
personnel, either military or civilian. The Chairman will
the ultimate responsibility for any products of the subgroup.
tional partitioning of a subgroup into special focus groups
be made, as appropriate. Attachment 1 identifies the par-
pating government agencies.

Meetings of the OSSWG will be held approximately every six
s, possibly supplemented by more frequent meetings of indi-
ial subgroups.

Approach

The optimum result would be identification of a set of
sting industry standards for operating system interfaces,
ices, and protocols that address existing and envisioned
iirements for Navy systems. However, neither the require-
s nor the possible solution set is obvious.

Consequently, the approach taken here will be to engage
istry in a dialogue that will result in articulation of both
Navy requirements and also current technology as regards
rating system interfaces, services, and protocols. It will
be necessary to identify a suitable process for evaluation
he marketplace in light of the requirements. To accomplish
, the OSSWG will establish three initial subgroups: a Re-
rements Team, a Current Technology Team, and an Approach
n. Figure 1 provides an overview of the work and inter-
ationships of these teams.



NOTES:

- * Responsible Team
- O Limited Involvement
- 1 OCD
- 2 Assessment
- 3 OSSWG Phase II Plan
- 4 Recommend Baseline Standards to SPAWAR 324
- 5 Prototyping
- 6 Revisions to Standards
- 7 1993 Draft Standards for OS interfaces, services, and protocols

Figure 1 - OSSWG Roadmap

Milestones and Deliverables

(R) = Requirements Subgroup
(AT)= Available Technology Subgroup
(AP)= Approach Subgroups

Products		Start	Deliver	
(R) Interim Req. Statement		now	3/1/89	
(R) Requirement Statement		4/1/89	6/1/89	
(R) "OCD"		6/1/89	6/1/90	(ver. 1.0)
Initial (amalgous of existing stds doc)			8/89	(ver. 0.1)
Interim (cleaned up, ready for feedback)			12/89	(ver. 0.2)
Draft			3/90	(ver. 0.3)
(AT) Draft Tech Report		now	3/1/89	
(AT) Final Draft Tech Report			3/9/89	
(All) Glossaary inputs to Jim O.		now	3/3/89	
(AT) Glossary			3/9/89	
(AT) Perform Evaluation		7/90	11/90	*Spec avail for prototy
(AP) POA&M	draft	now	3/1/89	
	final		3/9/89	
(AP) Abstract Model		now	7/89	
(AP) DID outline for OCD		now	6/89	
(AP) Evaluation Process	draft	now	11/89	
	final		6/90	

chment 1 - Government Participants

cy Abrams	Naval Air Test Center
Anderson	Naval Ocean Systems Center
Rick Barbour	Space & Naval Warfare Sys. Cmd.
n Bergman	Naval Ocean Systems Center
e Brouhard	Naval Ocean Systems Center
gory Bussiere	Naval Underwater Systems Center
no Cavallo	Naval Air Development Center
ve Cecil	Crane Naval Weapons Support Center
Conrad	Naval Underwater Systems Center

Linda Elderhorst	Naval Air Test Center
LT Karl S. Fairbanks, Jr	Naval Weapons Center
Manchi Gadbois	Naval Ocean Systems Center
Karen Gordon	Institute for Defense Analyses
Dan Green	Naval Surface Warfare Center - Dahlgren
Carl Hall	Naval Weapons Center
Steve Howell	Naval Surface Warfare Center
Phil Hwang	Naval Surface Warfare Center - White Oak
Russell Johnston	Naval Ocean Systems Center
Dan Juttelstad	Naval Underwater Systems Center
Ben Krug	Naval Avionics Center
Larry Lindley	Naval Avionics Center
Warren Loper	Naval Ocean Systems Center
John Machado	Space & Naval Warfare Sys. Cmd.
Tricia Oberndorf	Naval Air Development Center
Jim Oblinger	Naval Underwater Systems Center
James Reagan	Naval Surface Warfare Center - Dahlgren
George Robertson	Fleet Combat Direction Systems
Carl Schmiedekamp	Naval Air Development Center
Gail Sullivan	Naval Ocean Systems Center
Charlie Webster	Naval Air Development Center

DATA ITEM DESCRIPTION

FOR

OPERATIONAL CONCEPT DOCUMENT

FOR

NEXT GENERATION COMPUTING RESOURCES (NGCR)

OPERATING SYSTEMS STANDARD

VERSION 1.5

September 18, 1989

Approach Subgroup

Table of Contents

PREPARATION INSTRUCTIONS:

The Operational Concept Document describes the rationale for the Operating Systems Standard and its operational and support environments. It also describes the functions and characteristics of the standard in relationship to the Next Generation Computer Resources (NGCR) program.

The OCD represents a consensus among user agencies, industry and academia on the operational concept of the Operating Systems Standard.

Section/paragraph numbers in the OCD shall correspond to the numbers used in this DID.

Only cited documents shall appear in Section 2.

Section 4 shall define the requirements of the Operating Systems Standard. Only Section 4 shall use "shall" as defined in MIL-STD-490A, Section 3.2.3.6. In writing a requirement, there must be a method by which it can be tested.

Section 1

SCOPE

The Operational Concept Document (OCD) describes the mission of the Operating Systems Standard and its operational and support environments. It also describes the functions and characteristics of the standard in relationship to the Navy Next Generation Computer Resources (NGCR) program.

This section shall define the scope of the Next Generation Computing Resources (NGCR) Operating Systems Standard. It shall identify the schedule for introduction and use of the Operating Systems Standard. This section shall be numbered 1. and divided into the following paragraphs.

1.1 Identification

This section shall be numbered 1.1. This section shall contain a description of the NGCR program and the goals of this standards development.

This section shall begin with the following paragraph: "This Operational Concept Document describes the mission of the Next Generation Computer Resources (NGCR) Operating Systems Standard (OSS) and its operational and support environments. It also describes the required functions and characteristics of the OSS within the NGCR program."

This section shall further describe the relationship of the OCD to other documents produced by the OSSWG (i.e., describe the documentation tree).

1.2 Purpose

This section shall state the purpose of the Operating Systems Standard and identify its application domain.

1.3 Introduction

This section shall summarize the purpose and contents of the OCD. It shall contain a description of the need for a standard, an indication of the industry/academia/Navy collaboration in generating the

standard, and a roadmap to the document. The roadmap is not a duplicate of the table of contents. There should be descriptive sentences corresponding to the content of each section of the document.

The relevance of other existing standards and policies shall be discussed in this section.

Section 2

APPLICABLE DOCUMENTS

It is recommended that Mil-Std 490A, 483, and 962B be invoked in the writing of the Operating Systems Standard.

All documents listed here must be cited in the OCD.

Section 3

MISSION

This section shall describe the application of the standard. The rationale for a standard shall be provided. An overview of the NGCR Operating Systems Standards program shall be discussed.

3.1 Mission Need Requirements

This section shall provide a description of application domains (including operational platforms) to be supported by an Operating Systems Standard. Issues such as requirements for portability, reuseability, training, non-development items, (NDI) multilevel security, reliability, real time performance, Ada runtime support, and support for distributed systems shall be addressed.

Refer to documents (TADSTANDs, Navy Instructions, policy statements, and the like) in which requirements are stated.

3.2 Primary Mission

Sections 3.2 and 3.3 describe the justification for the standard. Based on an analysis of operational system requirements, a decision must be made as to primary and secondary missions.

An incomplete list of examples of possible primary and secondary missions might include:

- o User related support. (applications needs for OSS)
- o Support for Interface to other NGCR standards.
- o Ability to maximize use of NDI. Ability to define Navy requirements for industry in order to provide input to operating systems development groups.
- o Provide portability. Utilize state of practice technology at minimum cost for development. Maximize ability to upgrade systems in terms of cost and time.

- Provide a uniform base across all Navy systems. Promote ease of acquisition, maintenance, systems design, software development.

3.3 Secondary Mission

(See Primary Mission)

3.4 Operational Environment

Describe the intended approach to validating and using the Operating Systems Standard, the need for identifying agents responsible for establishing and executing relevant policy, the plan for maintenance of a library of accredited implementations, and other practical aspects related to selection and/or implementation of Operating Systems Standards. These issues must be discussed both with respect to new starts and upgrades to existing systems.

Describe the environment in which the standard will be applied and how it will be applied.

Describe support provided to Program Managers, Acquisition Managers, systems designers and developers, integrators, end users, operatins system software vendors and implementors, and others in operating system selection.

Support may include: evaluation criteria, trained personnel, reduced program cost, quality assurance, independent validation and verification.

Discuss the degree to which the Standard can be tailored to reflect various levels of complexity and functionality of operating system requirements that may be found in the application domain. Describe how an operating system can be qualified for a subset of the Standard's requirements.

Define classes of applications to be supported.

3.5 Support Environment

Describe the procedures that will be put in place during development and after deployment of the standard to maintain the Operating Systems Standard as a living document.

The configuration management process, revision mechanism, and acceptance procedures for operating system standard validation shall be defined.

Identify the agent responsible for maintenance of the Operating System Standard. Procedures for notification of the OSS maintainer of decisions by program managers to tailor or to seek waiver from the Standard should also be documented here. This will support appropriate evolution of the standard and facilitate the identification of "reusable tailorings."

This section also may be used to describe the process used to select/develop the Operating Systems Standard.

Section 4

OPERATING SYSTEMS STANDARD FUNCTIONS AND CHARACTERISTICS

This section shall describe the Operating Systems Standard functional areas in terms of the critical interfaces addressed. These critical interfaces are described in detail in the Abstract Operating Systems Model found in Section 10. The required services and protocols appropriate for each functional area are described here. These requirements will be derived from the operating system interface requirements defined by the requirements subgroup.

4.1 Operating System Standard Functional Areas

This section shall define the requirements for the functional areas of NGCR Operating Systems interfaces.

Subsection 4.1.x shall describe Interface Area x. The interfaces to be addressed are:

Ada Run-Time Environment Interface	(ARTEI)
Binary Application Program Interface	(BAPI)
Data Base Kernel Interface	(DBKI)
Local Device Interface	(LDI)
Graphics Kernel Interface	(GRKI)
Local Area Network Interface	(LANI)
Local Hardware Interface	(LHWI)
Local Processor Operating System (LPOS) to LPOS Interface	(OSOSI)
Man-Machine Interface	(MMI)
Source Application Program Interface	(SAPI)
Project Support Environment Interface	(PSEI)

Subsection 4.1.x.y shall describe service requirement y within Interface Area x. The requirements to be addressed are:

Language Support Services
Architecture Dependent Services
Capability and Security Areas
Data Base Services
Data Interchange Services
Event and Error Management Services
File Services
Generalized I/O Services
Man-Machine Interface (MMI) Services

Networks and Communications
Process Management Services
Project Support Environment Services
Reliability and Adaptability Services
Resource Management Services
Synchronization and Scheduling Services
System Initialization and Reinitialization Services
Time Services

4.2 Performance Levels for Operating Systems Standard Functions

This section shall describe the computer system configurations for each of the Operating Systems Standard performance levels. Each performance level will comprise a subset of the interface requirements defined in 4.1.

Performance levels will be a result of application system performance requirements.

The abstract model defines a preliminary set of performance levels.

Subsection 4.2.x shall describe a hardware configuration model as described in the abstract model.

Subsection 4.2.x.y shall describe a performance level for this configuration.

Section 5

GOVERNMENT AGENCIES

This section shall identify the standards development, support and user agencies within the major Navy organizations along with their responsibilities.

Section 6

NOTES

Section 10

APPENDIX

APPENDIX A

Glossary of Terms

A.1 Acronyms and Abbreviations

Appendix B

Abstract Model

Section 10

Appendix A

Glossary of Terms

10.1 Acronyms and Abbreviations

Appendix B

Abstract Model

Section 6

NOTES

NGCR

(Next Generation Computer Resources)

OSSWG

(Operating Systems Standards Working Group)

Reference Model
Version 1.02
1989 Aug 6

(printed 9/6/89)

Table of Contents

Table of Contents

List of Figures

OSSWG Reference Model for Embedded Operating Systems

Next Generation Computer Resources (NGCR)
Operating Systems Standards Working Group (OSSWG)
Approach Subgroup
Model Focus Group

The OSSWG Reference Model for embedded operating systems is a conceptual model which provides a context for the description of application developers' requirements, a context for the description and comparison of existing operating systems, and a framework for the specification of Operating Systems Standards (OSS) for embedded systems.*

1. Introduction

The OSSWG reference model is a conceptual model which provides a context for application program developer's requirements, for comparing existing operating systems and for standards specification. It provides a minimum, common set of conceptual embedded system building blocks with associated interfaces and functionality. Many of these system building blocks will be the results of other parts of the NGCR project. See Appendix A for background information on the NGCR project.

Consider the International Standards Organization (ISO) Open Systems Interconnect (OSI) reference model as an example of a reference model. That model defines seven protocol layers and associates each network communication function with one and only one layer. The model does not, however, specify a given protocol or protocol implementation for a given layer.

This OSSWG reference model is a model with the full embedded system as its scope, including some aspects of the project support environment. The model provides the basis for defining a set of concepts and conventions used by the system developer in designing and implementing the control system for the computer resources employed in the embedded system. Objectives of this model include supporting the portability and reusability of software components of the system and providing for the interoperability of software and hardware components. It will certainly allow for more compact and correct procurement specifications.

Many operating systems are built on a set of abstractions that make certain aspects of an application's execution "invisible" to the application itself. While this higher level view of the system can be valuable to the application programmer, this model will need to make some of those aspects visible so that the functionality of the operating system can be discussed and evaluated. Therefore the discussion of an operating system feature or aspect in the model does not necessarily imply that that feature should be easily or directly accessible to an application program.

The operating environment of operating systems built to the NGCR Operating Systems Standards will differ greatly depending upon the size and requirements of the system and its intended mission. It is expected that systems using an operating system compliant with

*Some of the text for this document is "reused" from a draft version of the POSIX System Architecture whose authors were Fritz Schultz, Jim Isaacs, Dale Harris, Sunhil Mehta, Al Weaver, Richard Scott and Doug Stevens.

the NGCR OS standards will not use all the features discussed here, or specified by OSSWG requirements documents, but will use tailored subsets for each particular application system.

A reference model must satisfy conflicting requirements similar to those encountered in more traditional modeling disciplines. The model must be structured enough to encourage the generation and use of standards and standard components. Yet it must also be flexible enough to accommodate tailored and special purpose components necessary to meet real-world needs. The reference model should also:

- be simple
- accommodate existing and imminent embedded systems standards, both hardware and software
- allow incorporation of both standards-based and proprietary subsystems
- reflect the full scope of application program developer's functional requirements
- allow system scaling
- accommodate new embedded system technology
- provide a means for comparing existing operating systems
- provide direction for future standardization and integration efforts

Note that the definition of this model is an engineering task and not a scientific one. There are many possible models, and while it might be interesting to contemplate an optimal one, an adequate solution is all that is required. Note that this model is intended to be conventional within computer science. The intention is not to break new ground, but to establish simple terminology and concepts for identification and resolution of architectural issues.

1.1. Multiple System Views

This model is actually composed of multiple views of embedded systems and their operating systems' interfaces. No one viewpoint seems sufficient to concisely describe the needs and goals of the NGCR OSSWG program.

First, in section 2., a model is developed which provides an overview of the system considered. The overview model defines system elements, to expose interfaces across which service requirements should be satisfied. The elements are chosen to expose those interfaces which are significant to the embedded system's developer.

Second, section 3. describes the important or critical interfaces between the embedded operating system and entities external to the operating system.

Section 4. then describes the basic services available across the interfaces of the operating system. The services are defined in a generic way, based on the model and current industry practice.

Section 5. finally discusses various application domains. This is a brief description of how the target systems differ with respect to several important requirements. The intent is to illustrate and bound the large variation that is expected among systems that may use the operating systems standards.

1.2. Abstraction and Levels of Aggregation

The model is described from the application developer's perspective, i.e. the model records the embedded system developer's perception (mental model) of the overall large distributed, embedded system and some aspects of its project support environment.

This point of view is used so that:

1. application developers will have the proper services to meet their requirements and
2. vendor implementation will not be constrained unnecessarily.

In addition to the application developer there is the application user and the system operator who will interact with an NGCR OSS compliant system. The application user is the end user of the system who may have no knowledge of the operating system interfaces used to provide the functionality seen when the system is used. The system operator, if there is one for a particular system, has special privileges and utilities to modify the system's database and configuration to meet changing needs of a mission. The system operator is expected to have some knowledge of the functions of the operating system and the structure of the particular application.

The developer's point of view has several levels of abstraction, with different objects being more significant at each level. For this model the five levels of abstraction are:

- System Design Level; which is concerned with the integration of multiple application programs into a cohesive system.
- Program Design Level; which is concerned with the integration of application modules and system services.
- Operating System Level which is concerned with the organization of and interfaces to the system services provided by the operating system. This is the SAPI (Source Application Program Interface) Level.
- Logical Device Level; which is concerned with the logical devices from which the system is composed.
- Physical Device Level; which is concerned with the physical devices from which the system is composed.

Figure 1-1 shows the system designer's viewpoint, where the major objects are applications and the system's devices. This view does not describe the system hardware. The system might represent one hardware processor or a large number of processors. At this level of abstraction the system is composed of one or more application programs which interact with each other, with the devices in the system and with the users via the User-Machine Interface (UMI).

Figure 1-2 shows a diagram of an application from the program design abstraction level. The internal boxes , each labeled "Service Group", represent modules providing system services. Those modules may be linked with the application code or may be part of the already resident operating system code. At this level the application designer sees the application as composed of application modules and OS modules which interact to perform the activities of the application.

The Source Application Program Interface level of abstraction focus on three entities: the application's code, the Source Application Program Interface (SAPI) and the operating system. The SAPI is the program designer's means to access the functions and objects of the operating system. The system's devices and other software entities are seen as being available through the services of the OS.

The Logical Device Level is concerned with the high level devices attached to or accessible to the system. At this level the particular characteristics of a specific device are hidden by its device driver and the operating system.

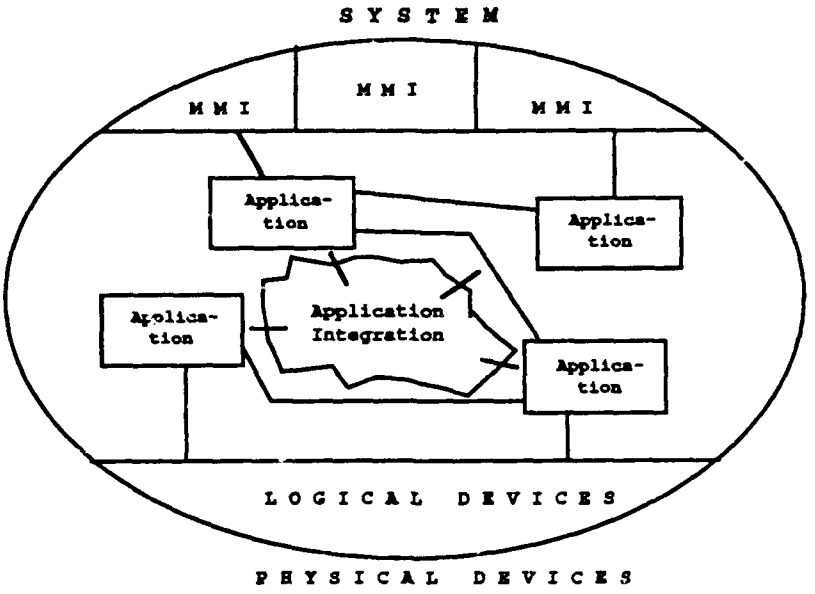


Figure 1-1
System Designer's View of the System

The Physical Device Level is the level seen by the writer of the device driver and by the application developer if the device needs to be accessed at a low, physical level for some special application.

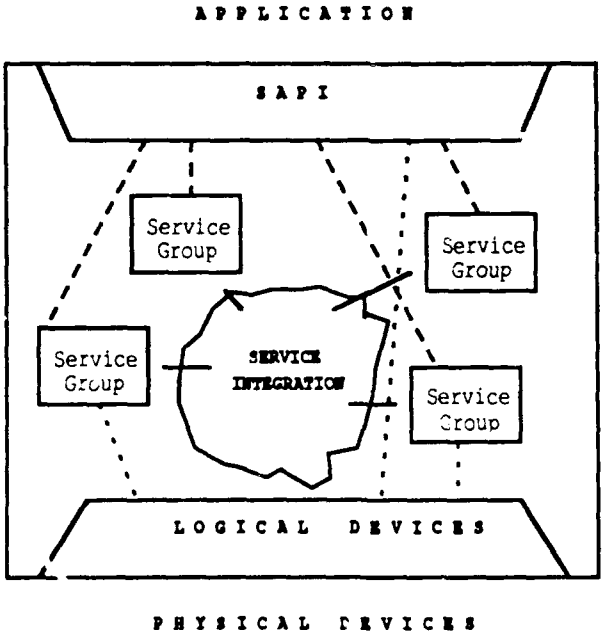


Figure 1-2
Application Programmer's View of System

2. System Overview Model

The full embedded system is represented by Figure 2-1.

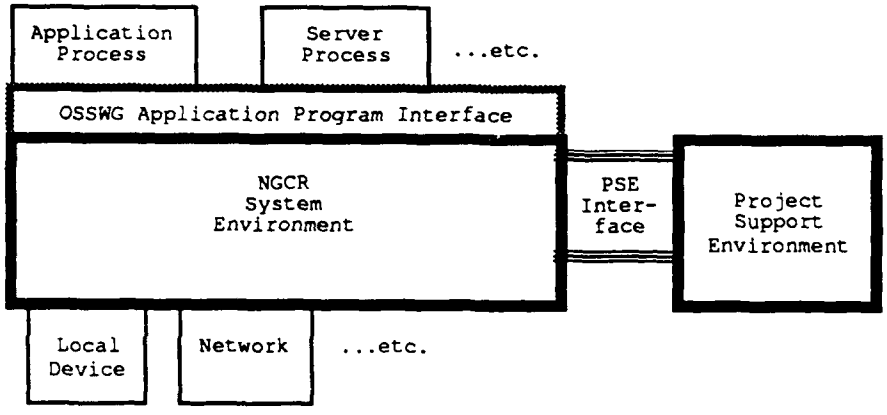


Figure 2-1
Local Processor Node Model from Applications Perspective

Figure 2-1 represents the embedded system as viewed by a system developer; this view corresponds to the program design level of abstraction. External features visible to the application developer include a variety of devices used to display and enter data. These devices include sensors and effectors which provide a means for the system to interact with the real world.

When an application is developed on the project support environment (PSE) the programmer accesses the application program interface (API) at the source code level through some particular language binding.

All run-time features can be conceptualized as being contained within the NGCR system block. All of the operating system features may be available locally or remotely. Some of the OS functions may be performed remotely if the system is a distributed system with multiple processor nodes. The term "processor node" is used to describe a single hardware subsystem which executes at any particular time, a single thread of control. A processor node may actually be constructed with multiple hardware processors, linked for self-checking and/or redundancy or may be a processor with auxiliary processors (such as floating point or I/O processors) attached. The aspect of the processor node that is singular is that it contains one and only one thread of control for its local operating system. Usually a processor node corresponds to one copy of an operating system but in systems with closely coupled processor nodes one copy of the operating system may support more than one processor node.

Two assumptions form the basis for the system overview model:

1. The application software system is represented as the execution of a collection of processes, where a process executes on a single processor node and contains a single, schedulable thread of control as seen by the local operating system.

2. Some mechanism for communications among these processes exists, whether communicating processes are located on the same or different nodes.

The following parts of section 2. describe the basic elements of the distributed OSSWG system and the relationships among them. This discussion defines the paradigm for the descriptions of services, interfaces and target domains which follow in sections 3 through 5.

2.1. Single Node/Application Model

Figure 2-2 identifies the major elements of a local processor node important to the embedded system developer and the relationships among them. While the Local Processor Operating System is shown as a single block its implementation is undefined by this model and it could be structured in many ways. For example it may very well consist of a proprietary OS, not compliant with the NGCR OSS, with specific NGCR OSS services and interfaces implemented "on-top-of" that proprietary OS.

Later this model will be integrated into a distributed environment, but initially the processor is defined as an isolated processor node. The elements include:

- Application Programs
- Local Processor Operating System (LPOS)
- Application Program Interface (API)
- Local Devices
- Project Support Environment (PSE)

One or more application processes may run on the processor simultaneously, as represented by the process rectangles at the top of the figure. The applications run as independent software entities and communicate among themselves via a variety of communications mechanisms provided or managed by the local processor operating system.

The applications make use of devices attached to the local processor to perform a wide variety of actions. These local devices are represented along the bottom edge of the figure, and they
include sensors, effectors, networks and direct connections to other computing systems.

The Local Processor Operating System (LPOS) allocates the shared local devices among the applications competing for these resources. Processor time, memory, and other finite processor resources are also shared among the applications, mediated by the operating system. The LPOS also supports communication and cooperation with other LPOSs at other processor nodes of the system.

The block labeled LPOS in the figure actually contains only the run-time elements of the operating system which usually run in supervisor mode or protected mode. These elements of the LPOS are the parts that handle system service requests from the application programs. Other parts of the LPOS may run as server processes or as library routines linked with application programs. These server processes may have special authorizations or

capabilities but are scheduled and serviced by the run-time elements of the LPOS in the same manner as user processes.

The LPOS also supports an Ada Run-Time Environment (ARTE). The Source Application Program Interface (SAPI), the compiler vendor supplied compilation library and interface code generated by the compiler together form the complete ARTE. A good interface standard should not depend on whether the ARTE functions are implemented in the low level run-time elements of the OS or in code from the Ada compilation system. While it is expected that the OS will support the needs of programs written in Ada, other languages and their run-time support will also be needed for particular projects.

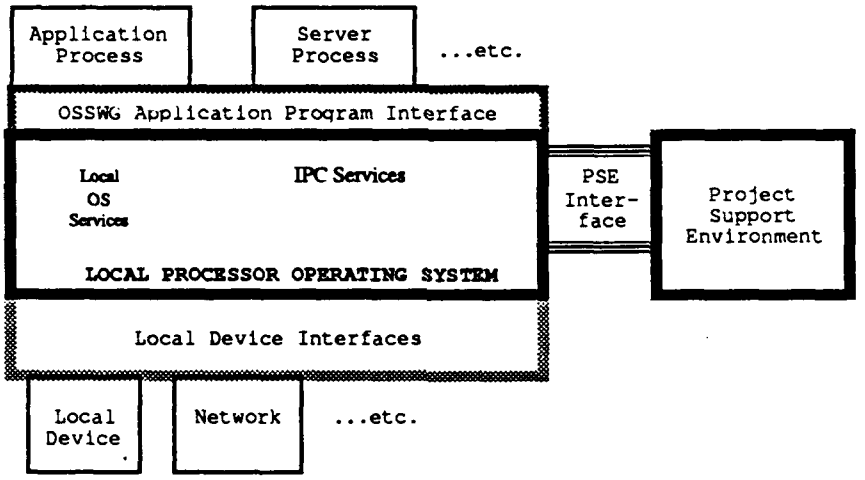


Figure 2-2
Local Processor Node Model from OS Perspective

In order for the OS to protect system integrity and ensure system database consistency, applications competing for system resources must access all system resources via system service requests. The formal definition of these requests (or system calls) defines an Application Program Interface (API). The API will specify a sufficient interface between the application program and the underlying operating system and includes the operating system services which are described in section 4.

The API has several different representations. One set of representations is represented by the Source Application Program Interface (SAPI) which is a programming language binding to the API for some particular source language. For each language allowing service requests to the operating system there is a Source Application Program Interface. The SAPI is a set of subprogram calls to be invoked to access operating system services. This is the representation used by the programmer and is a primary interface used in the PSE. Another representation is the Binary Application Program Interface (BAPI) which is the calling mechanism used by the compiled code to access the operating system routines which are not part of the application code. The name **Binary** Application Program Interfaces is meant to imply that this interface is at the machine code level. The BAPI will in most cases be LPOS implementation and processor dependent, but might be standardized for a single processor type or a family of processors to allow some degree of portability of compiled code across different implementations of the LPOS. Each of the critical operating system interfaces is listed and briefly discussed in Section 3.

The Project Support Environment Interface (PSEI) is the block on the diagram between the LPOS and the PSE. This interface allows communication between the PSE and the LPOS in a development environment. This interface may not be subject to standardization at this time, depending on an analysis of the risks and benefits of such a standard PSEI. The PSEI provides the following services:

- Down-Loading of compiled programs
- Up-Loading of execution-time debugging information
- Remote Control of the embedded system by the user of the PSE, including the execution of debuggers running partially on the PSE an partially on the target system.

Note that these kinds of operations are similar to those required for coordination of user programs running on multiple, distributed processors except that very detailed knowledge of the execution environment on the target will need to be communicated back to the PSE. Most of the functionality of this interface may be available from the SAPI/BAPI.

The Local Device Interfaces block in the figure is the set of device drivers used by the LPOS to access the different devices. The interface between the device drivers and the LPOS is the Local Device Interface (LDI).

The most important interface, for OS standardization, is the SAPI. With a standard Source Application Program Interface an application routine can be transported to a new target system by recompiling the source code. A standard SAPI will also support interoperability and software reuse at both the subprogram and subsystem level.

A development version of the embedded system may have a Real-Time Non-Intrusive (RTNI) testing device attached to it for monitoring the system's performance and debug the application (and perhaps the system) software. An RTNI device is not shown in the model because a truly non-intrusive device will not be "visible" to the software or to the operating system of the tested system. An RTNI device may under some conditions or modes (e.g. during test setup) be visible to the system as a special purpose device and an NGCR OSS compliant operating system should be capable of communication with such a "visible" device. The RTNI system may itself contain a computer system that uses an NGCR OSS compliant operating system to run applications that collect data about the tested system.

2.2. Network/Application Model

We now expand the system to expose network-related interfaces. Setting aside the local node model for the moment, figure 2-3 relates the application processes to a conceptual model based upon the OSI reference model for network services.*

Applications gain direct access to the network services at levels 4 through 6 via service requests specified in the API. The network is just another system resource allocated among the competing processes, although it is an important one. Both connection-oriented and connectionless data transport services may be available.

In some cases, it may be desirable for processes located on the same node to communicate via network services. For example, if it is possible that one of the communicating pro-

*A detailed tutorial on the OSI Reference Model is available in "ISO 7409 - Open Systems Interconnection Model - Basic Reference Model". Note that the OSI layer 7 (Application Layer) represents, in the Network/application Model, the applications running above the OSSWG API.

cesses could potentially be moved to a different processor node, the use of network services makes sense. If the processes will always be located on the same node, other methods may improve performance. This is an application-level architectural decision that has substantial impact on design and implementation of distributed system applications. The actual source level interface for local and remote communication between processes may be the same with the responsibility falling to the OSs and/or compilation systems to determine the location of a process and facilitate the proper communications.

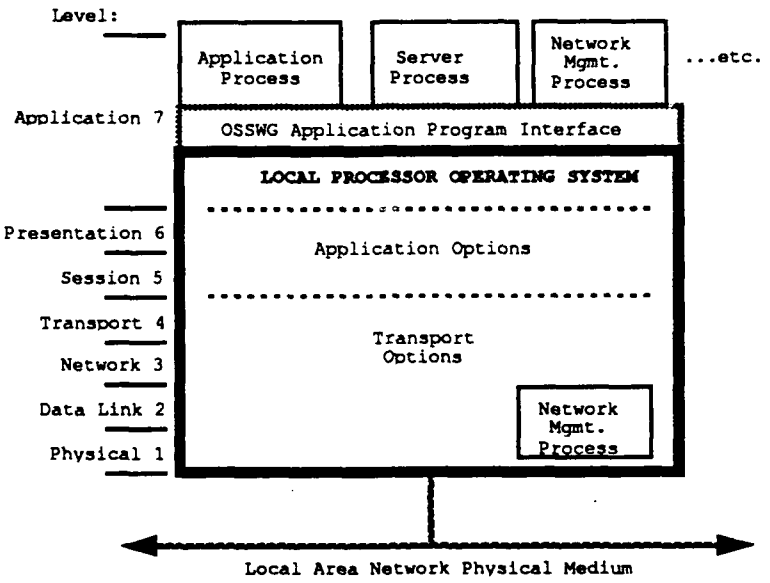


Figure 2-3
Network Model from Applications Perspective

2.3. Network Communication Model

Figure 2-4 integrates the node/application and network/application models into an element of a local area network coupled distributed system.

No new elements are introduced in this integrated model. A major feature of this model is the integration of the network protocols with the operating system on the local node. Note that the upper network protocol layers are closely associated with the operating system. This is driven by the fact that the session layers provide communications services among processes. The process is an operating system construct and is managed by the operating system, while the session and datagram services are network constructs and are managed by the protocol software. This requires close coordination and integration between these software elements. This can be a major source of difficulty during development, integration, and operations.

Note that the application process must pass service requests to the operating system via the API to gain access to network services. As discussed above, the API provides data communications transparency to the applications. This means that the complexity of the network is hidden from the applications behind the API.

Network management functions may be associated with any network layer. Figure 2-4 shows network management processes associated with the upper protocol layers, as well as with the lower layers. This is due to the fact that the upper layers are closely associated with the operating system and may use operating system services to perform network management functions. The lower layers, however, are more closely associated with the physical media and may not have direct access to the processor.

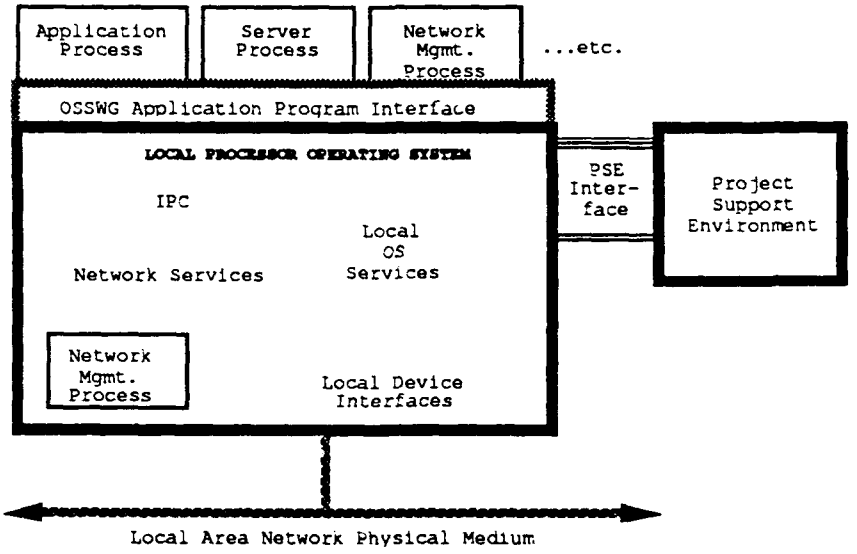


Figure 2-4
Integrated System Node

Often the operating system allows the application programmer to refer to entities attached to the local network by logical names rather than by network address. That capability can be very useful for separating the specification of the location of an entity from the code that uses it, however some applications will need to refer to network entities by their physical address.

2.4. Program Distribution

Many Navy computer languages do not have any support for concurrency but rather depend upon the facilities provided by the operating system via calls to OS services. The Ada language is one exception because it has language level support for concurrency, other exceptions are languages designed for signal processing. With respect to programs written in languages that support concurrency, there are two levels of concurrency. Either level (or both) may be mapped to OS processes. The first is the Ada **task** level where each unit corresponds to an Ada main subprogram or an Ada Task. Concurrency at this level considers the relative priorities and scheduling of Ada tasks within the program and their communication via language constructs. The second level of concurrency, the **program** level, corresponds to a single Ada program together with all of its dependent tasks. Concurrency at this level considers the relative importance of the individual programs in the system competing for system resources. Also program-to-program communication is via

the operating system or shared data because the Ada language provides no communication facilities at this level outside of file input-output.

When an application is to be distributed across multiple processor nodes that may not share common memory, there are several ways to partition Ada programs. Two of the partitioning methods correspond to the levels of concurrency above. A program can be distributed at the Ada task level with the OS, compilation system or the developer deciding where each task is to execute. Distribution at this level would require communication across processor nodes with full Ada tasking semantics. Distribution at the program level means that a program and all of its tasks execute on a single processor node. If distributed processing is needed then the application developer must divide the application into separate programs that communicate via calls to the communication facilities of the OS. This is current practice in real-time systems including Ada based systems.

Distribution can also be at the "virtual node" level. A virtual node consists of a collection of related Ada library units. A program may be one virtual node, which corresponds to the program level distribution or a program may be composed of several virtual nodes which together contain all the library units of the program. A virtual node is assigned, by the developer or the OS, to a particular processor node. More than one virtual node of a program can be executing on the same processor node, but a virtual node may execute on only one processor node at one time.

2.5. System Resource Allocation Executive(SRAX)

Conceptually the SRAX is the single operating system for the whole system if one exists; it manages systems resources across processors so that to the applications developer of the system, or some aspects of it, appear to be controlled by a single centralized operating system. The SRAX primarily is responsible for scheduling and allocating resources that affect more than one local processor node, but since most resources are local to some local processor node the SRAX must cooperate with or control the local scheduling mechanisms. Note that the coordination across local processor nodes includes nodes that have different types of processors. The communication between the different Local Processor Operating Systems requires a set of functions and protocols that may not be part of the Application Program Interface.

A platform may have several SRAX-level clusters of local processor nodes that share a communication network but do not cooperate with each other for resource sharing at the operating system level. Effectively this would provide multiple SRAX systems that can communicate across a network. Also on the network may be simple independent systems that may not need any inter-processor coordination and only have the LPOS part of an operating system with no SRAX level OS.

Figure 2-5 shows two nodes of a multi-node system. Many embedded applications require that the nodes of the system be able to coordinate services and resources. This can be done all at the application level with the application programs communicating with each other and then with their individual LPOS which will be fully supported by the API; such an application system would have no need for SRAX level services. The LPOSs could also communicate and coordinate resource usage directly via the SRAX which may provide such services as dynamic load leveling and automatic reconfiguration.

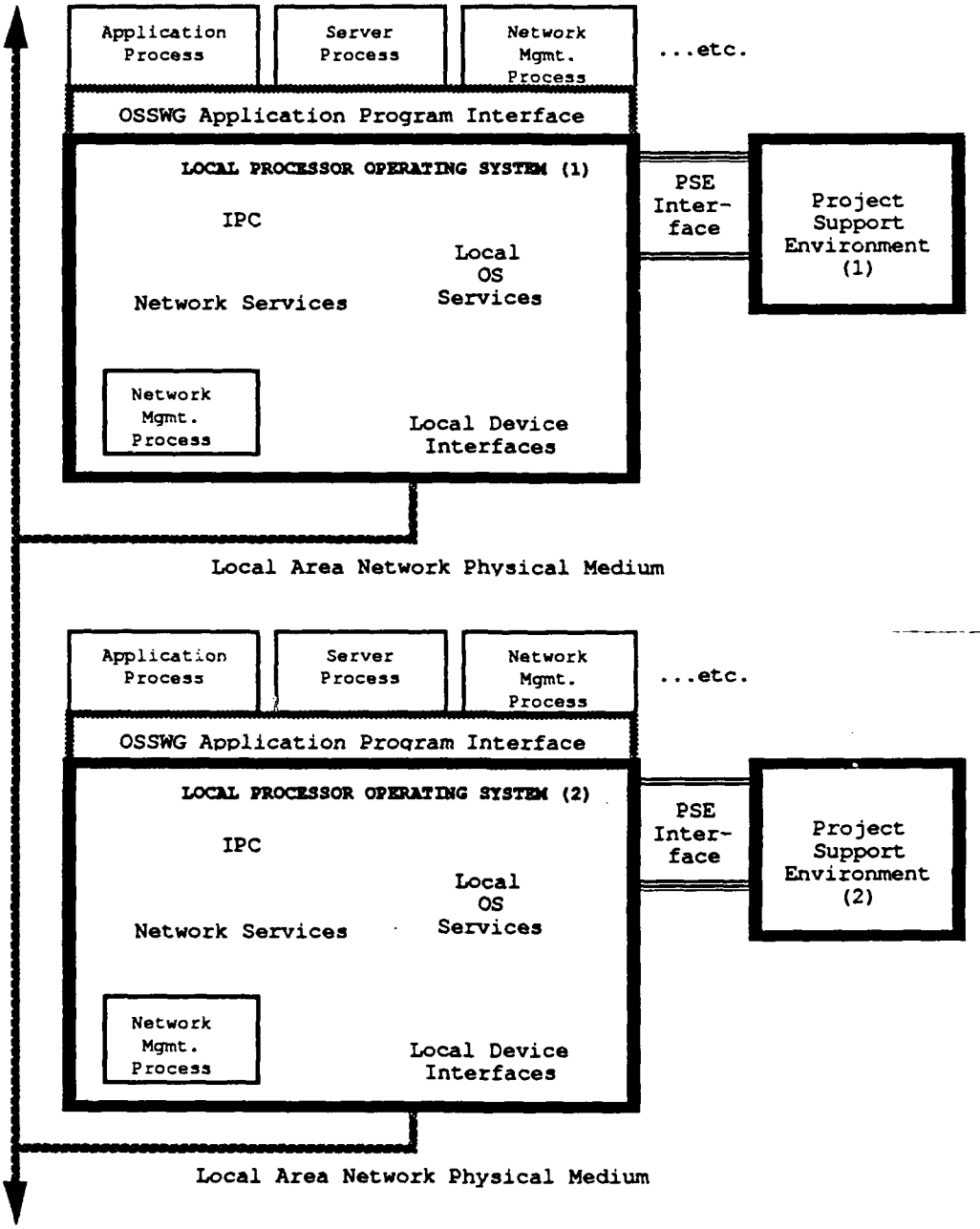


Figure 2-5
Distributed System Nodes

There are many ways that the SRAX could be implemented.It could be implemented as a single centralized program running on a single processor. It could be implemented in a fully distributed manner such that every local processor node has its own portion of the SRAX program Parts of the SRAX could even be implemented in hardware for high performance. The SRAX could be implemented in an hierarchical manner with higher level SRAX components doing more global coordination of resources while lower level SRAX components coordinate the use of resources that are more local or that require faster response times. Realistic implementations would probably have some services centralized

and others distributed on some or all of the local processor nodes. Even with a centralized implementation there may be one or more "backup" processors that could take over execution of the centralized part of the SRAX if the initial one fails. Figure 2-6 is a diagram of the different parts of the operating system and their location for one simple implementation scheme. The SRAX in the figure has two components: the centralized part (SRAX - C) and the local part (SRAX - L). The centralized part executes on one processor and coordinates the other processors, while the local part of the SRAX and the individual LPOS parts execute individually on each processor.

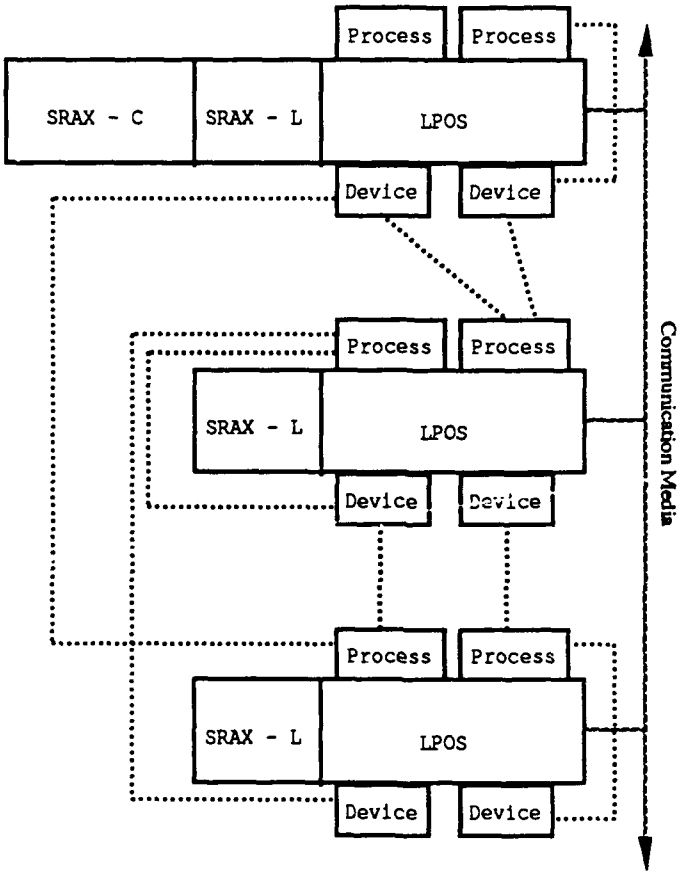


Figure 2-6
LPOS and SRAX parts of the Operating System

The different parts of the SRAX communicate with each other to schedule the resources of the system. It is this communication which makes system level control possible but also can potentially cause a severe communication load on the system.

3. Critical Interfaces View

This section discusses Operating System interfaces that are necessary to meet the goals of the NGCR project. These interfaces are defined by the entities that the OS and application programs must interact with (see Figure 3-1.). These interfaces arise from the need for the different components to be integrated into a cooperating system. Not all of the interfaces are visible to the applications developer and not all need to be standardized, but all of the interfaces will be present in one or more implementations. The interfaces may be provided by compiler or operating system vendors or by the application programmer rather than being part of the NGCR OSS or any other NGCR standards.

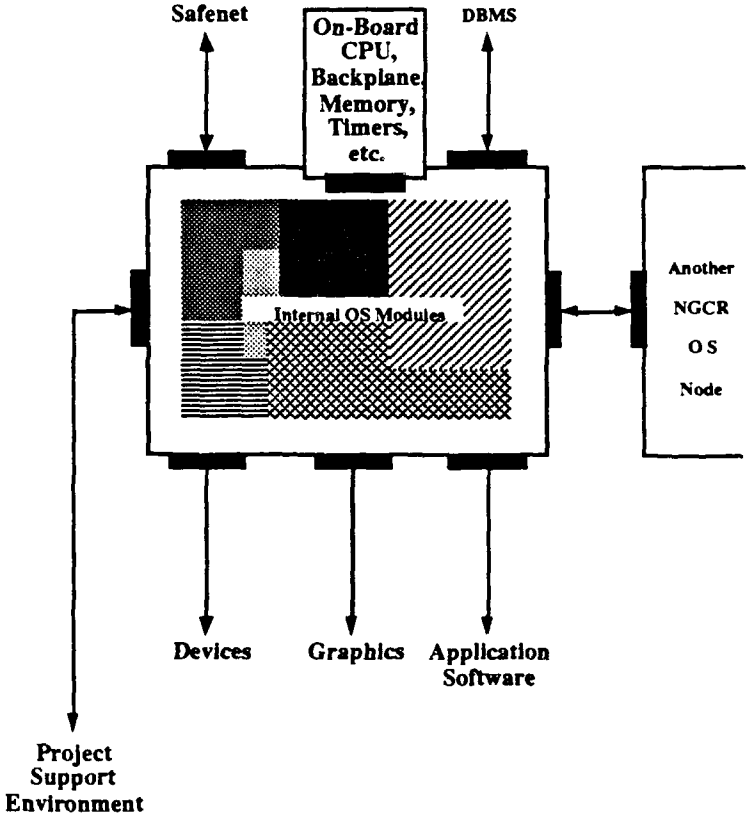


Figure 3-1
Entities that Interact with the OS

The viewpoint taken is that of the Local Processor Operating System (LPOS), i.e., the component of the total operating system that executes on a local processor node. An LPOS may communicate with other LPOS instances via the backplane, via a local area network and via shared memory. The following interfaces are discussed:

(ARTEI) Ada Run-Time Environment Interface
(BAPI) Binary Application Program Interface
(DBKI) Data Base Kernel Interface
(GRKI) Graphics Kernel Interface
(LANI) Local Area Network Interface
(LDI) Local Device Interface
(LHWI) Local Hardware Interface
(LSCI) LPOS — SRAX Coordination Interface
(OSOSI) LPOS to LPOS Interface
(PSEI) Project Support Environment Interface
(SAPI) Source Application Program Interface
(UMI) User-Machine Interface

3.1. Ada Run-Time Environment Interface (ARTEI)

(Ada Run Time Environment <-> Application Software)

This is the interface between the application Ada programs and the run-time environment required by the Ada programming language. Figure 3-2 shows that the ARTE may be implemented in three parts, part loaded with each application program from the Ada Run-Time library provided by the compiler vendor, part which is generated by the Ada compiler during the translation of the Ada program and part which is a subset of the OS interfaces. It is an issue as to how much of the necessary ARTE is part of the operating system and how much should be provided by the compilation system of a compiler vendor. Most likely some parts of this interface will be provided to allow applications to modify or "tune" the run-time system and other parts of the interface will only be used by compiler generated code. Standardization of the interface to the OS part of the Ada services will allow the efficient integration of the Ada Run-Time Environment and the LPOS. This interface is one aspect of the more general High Order Language Binding Interfaces which may include language bindings and run-time support for such Navy languages as CMS-2 and Lisp.

3.2. Binary Application Program Interface (BAPI)

(Binary Application Software <-> LPOS)

This interface is the machine code level calling sequences to the LPOS. It is the binary level version of the SAPI. This is logically the same interface as the Source Application Program Interface described below but may require a separate standard if the compiler vendors are to be decoupled from the LPOS vendors. As an example of a BAPI, some machines use a set of software interrupt instructions to call operating system routines; the specific interrupt numbers and the specific conventions for parameter passing are part of the BAPI. Without a common interface at the binary level a separate version of each compiler (or at least its code generator) may be required for each different implementation of the LPOS software, and code compiled by different compilers may not operate together. Note that even with standardization of the BAPI, we have at best a set of standards, one for each processor or processor family.

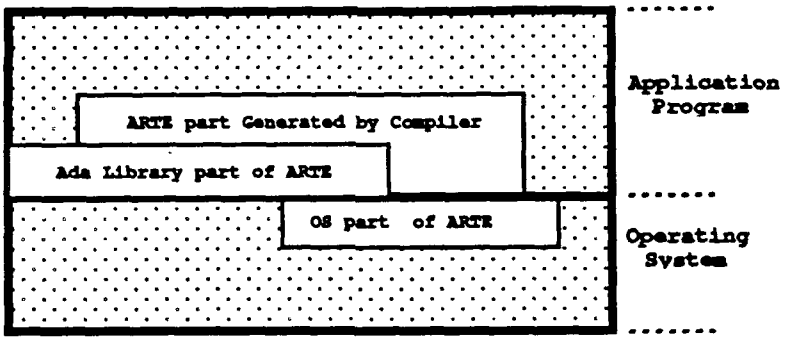


Figure 3-2
Parts of the Ada Run-Time Environment (ARTE)

3.3. Data Base Kernel Interface (DBKI)

(Data Base Kernel <-> Data Base Management System)

Those systems which will require a multi-level secure OS will require that some low level parts of system level components (LPOS, DBMS etc.). be fully integrated (i.e. be part of the TCB); therefore the Data Base Management System (DBMS) must have efficient access to the TCB. If the TCB is integrated with low level parts of the OS then those OS components must provide the needed interfaces for the DBMS The DBKI is the interface that the DBMS system uses to access the TCB or specialized OS components. Without a DBKI built into the LPOS a DBMS may have to be built "on top of" the OS, which could lead to poor DBMS performance. If the data base kernel interface is of general usefulness to application programs, then this interface may simply be a subset of the application program interface (SAPI and BAPI) to the operating system and therefore not a separate interface. It could also be a separate, special purpose interface, that is provided only for those systems that use a DBMS.

3.4. Graphics Kernel Interface (GRKI)

(Graphics Language <-> LPOS)

This interface is listed because the NGCR Graphics Language/Interface may require specialized low-level OS functions for performance reasons. If special services are needed, then this interface will be required. It is not clear at this point whether the application program interface (API) will be sufficient for the needs of the Graphics Language/Interface implementation.

3.5. Local Area Network Interface (LANI)

(Local Area Network <-> LPOS)

This interface separates the LPOS software from the local area network software/hardware subsystem. The NGCR SAFENET effort may provide the interface or a set of interfaces

that the LPOS software can use. If the SAFENET standards do not provide usable interfaces then the OSSWG OSS will need to define these important interfaces.

3.6. Local Device Interface (LDI)
(Local Devices <-> LPOS)

The LPOS software can be written with device drivers included for the specific system devices, but a more modular system would be possible if there were standard device driver interfaces. Device driver interfaces make it easier to add new devices to the system or to rearrange the configuration of devices. The LDI can be at two levels: at the LPOS to device level if the device drivers are custom built into the LPOS or at the LPOS to device driver level if a device driver interface is defined for the LPOS. Defining a device driver interface allows a new device to be added to an existing OS by adding a new device driver rather than requiring a new version or modification of the OS.

3.7. Local Hardware Interface (LHWI)
(Local Hardware <-> LPOS)

This interface is usually hidden (i.e., proprietary to the LPOS vendor). This interface is hardware dependent and should probably not be standardized by the NGCR effort. This interface includes as a subset the interface to the backplane. While the backplane hardware itself will be built to NGCR standards there may not be a standard, board-level, hardware interface to the backplane. This interface to the backplane and other local hardware will then depend on the design of a particular board. The CSR standardization effort, which is affiliated with the backplane effort, is standardizing some of the board-level register assignments but will most likely leave a lot of room for hardware specific register assignments.

3.8. LPOS — SRAX Coordination Interface (LSCI)
(LPOS <-> SRAX)

This 'interface' is primarily a set of protocols, data formats and conventions that provide communication between an LPOS and the SRAX. The functionality provided by this interface, to a large degree, determines the amount of coordination of LPOSs available to the SRAX component. If there are more than two levels in the operating system hierarchy then there may be the need for multiple interfaces. This interface may be hidden (i.e., proprietary to the operating system vendor) in a system where both the LPOS and SRAX are developed as a single system.

3.9. LPOS to LPOS Interface (OSOSI)
(LPOS <-> LPOS)

This interface allows one LPOS to communicate with other LPOS instances in the system and allows instances to share resources and to cooperate with each other. This interface is needed if the goals of reliability and dynamic reconfiguration in a heterogeneous system are

to be provided by the operating system rather than being available only if supplied by the application software. The scope of the services provided by this interface depends on the level of coordination needed among the separate processor nodes. Note that these interfaces are not application program interfaces although the OSOSI may be necessary to support the functionality provided by some parts of the application program interfaces.

An example of the type of communication needed is when two LPOS nodes share a memory board across the backplane the two nodes must coordinate with each other to allocate and use the memory. Without an OSOSI the application developer will have to perform all the memory management coordination in the application. Another example of the kinds of messages needed are those to coordinate the live insertion or removal of processor boards in a running system as supported by the NGCR backplane.

3.10. Project Support Environment Interface (PSEI)
(Project Support Environment <-> LPOS)

This interface provides a means for the PSE to interact with the LPOS for loading software, testing, debugging etc. In many systems this interface would be removed before the system became operational. The standardization of this interface will make it easier to have a common PSE for different LPOS instances. Some systems will have a different, hardware based, non-intrusive testing interface to the LPOS. The non-intrusive testing interface is by its design invisible to the LPOS hardware and software and therefore is not important to the NGCR OS Standards effort even if it is of extreme importance to a particular project.

3.11. Source Application Program Interface (SAPI)
(Source Application Software <-> LPOS)

This interface is the one that is normally thought of when OS interfaces are being discussed. This is the interface (or set of interfaces) which the applications programmer uses to develop embedded systems. This is the high order language bindings (Ada etc.) to the OS system calls. The BAPI is the compiled, binary version of this interface.

3.12. User-Machine Interface (UMI)
(User <-> Application)

The User-Machine Interface is the interface between the application user and the application programs. The hardware used for this interface is often some combination of special purpose display devices and user input devices. This interface can be considered a special form of device interface for which a standard set of device driver commands would be useful to promote software transportability and easy movement of development engineers among projects and users among embedded systems. Any standardization at this level may come out of the NGCR Graphics Language/Interface working group. Many existing operating systems have very little support for the UMI except for perhaps a command line parser

4. Operating System Services

This section describes the major groups of operating system services that may be required of the NGCR OS. Not all of these services require a programming interface; therefore we can describe the services as either explicit or implicit services. Explicit services are those that can be accessed from an application program (via the API) and generally are only provided when requested. Implicit services, on the other hand, are services that the OS provides without a direct request. An example of an implicit service is the prevention of one program from writing over the memory of another. An example of an explicit service is a call to an OS routine to output a block of memory to some device.

The OS services often are available at or support more than one of the interfaces described in section 3. For each of the services in this section is listed the associated interfaces.

4.1. Language Support Services

Service's Interfaces: ARTEI

Navy languages and their standard libraries have specific OS needs that the NGCR OSS should meet. This section emphasizes the needs for Ada support because that language is currently required for all new weapon systems and major modifications to weapon system programs.

4.1.1. Ada Language Support Services

These services support the use of the Ada programming language. This section describes some special needs of the language which may be addressed by services within the system.

4.1.1.1. Full Ada Language Support

While an NGCR OSS compliant operating system may be implemented in various languages it should support the execution of programs written in Ada. At the least this means that the operating system together with the compiler's run-time library should include all necessary parts of an Ada Run Time Environment (ARTE).

For highest efficiency some parts of the ARTE should be an integral part of the operating system although the interface definition itself need not depend on that integration. An example of an integration problem is that of task scheduling. Many current implementations of an ARTE "on top of" an existing operating system schedule Ada programs as single entities. If an Ada task is running and becomes blocked, the OS does not consider other tasks in that program for execution, but only considers other programs.

4.1.1.2. Exception Propagation to OS

When an exception from an application or hardware event is propagated to a operating system this service handles the communication of the event to the proper application routine.

4.1.1.3. Interrupt to Task Mapping

When an interrupt occurs, this service will ensure the correct mapping from interrupt to Ada task is made (even when the interrupt and tasks are located on separate processor nodes of an SRAX level distributed operating system).

4.1.1.4. Priority

These services support the full priority semantics of the Ada tasks.

4.1.1.5. Rendezvous

These services support the rendezvous of tasks (from tasks being implemented as within one single process to tasks being implemented as distributed processes).

4.1.2. Support for Other Languages

Other languages have fewer requirements on the OS than Ada. The C language itself places almost no requirements on the OS. The usual C language libraries, however, require simple services like byte stream I/O and the ability to create and receive signals. The CMS-2 language also has few requirements on the OS. Lisp requires support for garbage collection which can be provided by the hardware and the OS or provided by the Lisp Run-Time system.

4.2. Architecture Dependent Services

Service's Interfaces: LDI, LHWI, LANI, GRKI, OSOSI

These services allow the system to interface with non-NGCR resources such as computers, networks, and operating systems. This will facilitate portability, technology insertion and the reality that the NGCR system will need to interface with many existing systems which will be in use for many decades to come. These services may allow the system to interface with today's Navy standard computers such as the AN/UYK-44 or AN/UYK-43 as well as special purpose computers such as the AN/UYS-1. This set of services also includes means to access the special hardware features of a system and is in effect a standard way to access non-standard functions

4.3. Capability and Security Services

Service's Interfaces: all interfaces

These services support the ability of the system to control usage such that system integrity is protected from inadvertent or malicious misuse. These protection services provide a mechanism for the enforcement of the policies governing resource usage. Note that many of the security services are implicit services, i.e., they are provided without an explicit request to the operating system. There are two distinct classes of system access with which operating system services must be concerned: physical access and logical access.

Security services at the physical level are used to protect against security compromise, given unauthorized personnel may have physical access to system hardware. Typically, the physical access is to a terminal and/or terminal/display cables; however, physical access may also include network cables, central processing units, disk drives or tape drives. Different types of physical access by unauthorized personnel may require different operating system services and/or hardware to support secure operation. For example, if unauthorized personnel have physical access to the network cable, then services may be needed to support the encryption of any secure data passing through the network. This is because a per-

son may hook a data reading device to the cable without needing to get access via the operating system.

Logical access is the ability to interact with the operating system via a terminal/display. Security services at the logical level can be implemented through passwords and watchdog timers.

Capability services attach operation lists which limit functions' (processes') ability to act on resource objects. This is to ensure the resources are not misused. Access to resources can be protected by services using capability lists as well as access lists, lock/key mechanisms, global tables or through dynamic protection structures services.

4.3.1. Prevention of Unauthorized Access

Access to the system may need to be guarded from attempted access by unauthorized personnel. The points of access to the operating system which we typically are concerned with are through the SAPI or PSEI. Given the mode of operation (system high, multi-level, open) at which the system is operating, these services differ and have differing implications on other system services (such as reliability, naming etc.) and system performance.

4.3.2. Prevention of Data Compromise

These services prevent access of data by users not authorized to the data. These services may be implemented using access lists on files (and directories) and/or encryption of data or in other ways.

4.3.3. Prevention of Service Denial

These services ensure that a service request will be met by the operating system in a reasonable time if the requestor is authorized to use the service. These services ensure that a bandit user or process cannot cause system malfunction by monopolizing system services or resources.

4.3.4. .i.Security Administration;:.

This category involves services to allow the managing of the security system including the administration of permissions to personnel, data, and services as well as capability lists. In addition, it permits the administration access mechanisms (passwords or whatever) and services which allow the system to switch modes of operation. The services will likely be accessed by the system operator with security responsibilities through the system operator services.

4.4. Data Base Services

The database management system in an embedded system has several functions, including access control, consistency checks, maintaining consistent copies for fault-tolerance and security. The need for Data Base Services as part of the OS arises because of the interaction of the DBMS's need for performance and multi-level security needs. If parts of the OS are part of a Trusted Computing Base (TCB) for a multi-level secure system, then the lower level parts of the database management system (the "database kernel") will have to be part of that TCB or be built "on top of" the OS. The Data Base Services may be specialized services for use to support a DBMS or they may be of general use for application programs. The NGCR Data Base Working Group has the responsibility for defining interfaces for a full DBMS.

4.5. Data Interchange Services

Service's Interfaces: OSOSI, SAPI

This set of services provides data conversion among different data representations. One scheme for providing these services is to have a single canonical representation for the important data types (integer, real, time etc.) and then each implementation of the OS or compiler would provide conversion functions between the canonical representations and its own internal representations.

4.6. Event and Error Management Services

Service's Interfaces: ARTEI, DBKI, GRKI, LANI, LDI, LHWI, UMI, OSOSI, PSEI, SAPI

These services provide a common facility for the generation and communication of asynchronous events among the system and application programs. A major use of the event services is to report error conditions, but they may be used by device drivers and the OS to provide an indication of some condition to the application programs.

4.7. File Services

Service's Interfaces: BAPI, DBKI, LDI, LANI, OSOSI, SAPI

These services allow the system and applications to create permanent storage locations for data. The data is stored on files and the files are organized in directories. Files are managed and accessed through logical names by the many system components that use the files, such as the application, system operator, and program support environment.

4.7.1. Naming and Directory Services

These services allow the access of files and directories through logical names rather than the actual hardware device naming conventions. The services may allow sharing of files at various levels. For example, the services may not allow any shared naming of files and directories between systems, or they may allow shared files by explicit naming, or they may allow shared files by implicit naming. The directory services present a view or views of the directory structure to the application or system operator.

4.7.2. Real-time Files

Real-time systems often need special files to ensure fast, predictable and consistent performance in time critical situations. The need for a known response time for a given I/O function drives the design of these files and services. One service may preallocate the complete disk space needed for a file at creation time, while another guarantees that records within files are aligned in an optimal way (such as along word boundaries). Services may support the access of records within the file in ways that make response time constant or bounded, such as direct access.

4.7.3. File Modification Primitives

Primitive services for files and directories include the ability to read a portion of the file, write to a portion of the file, open access to a file, create a new file, close access to a file, and delete a file. These services may be very complex. For example, the access to read or write may be direct (by record number), sequential (one record at a time) or indexed (by a key).

In addition, services may be needed to support the merging, appending, splitting, and copying of files. The services may need to support a variety of file structures such as linked, segmented, contiguous, serial or directory.

4.7.4. File Support Services

Additional services support the physical devices on which the files and directory reside. These services include the dismounting/mounting of medium, the formatting of medium, and the partitioning of media.

4.8. Generalized I/O Services

Service's Interfaces: DBKI, LANI, LDI, LHWI, UMI, OSOSI, SAPI

Generalized I/O services provide higher level constructs and functions for doing I/O to devices that do not fit well into the common file I/O paradigm. These services include non-blocking I/O and I/O to special devices. In non-blocking I/O output or input is initiated under program control but the program continues execution while the transfer takes place. Many special hardware devices may need I/O supervised by the operating system.

4.9. Graphics Kernel Services

Service's Interfaces: GRKI

This interface provides low-level access to graphics services. The NGCR Graphics Language/Interface may require low-level access for performance reasons. It is not clear at this point whether the application program interface (API) will be sufficient for the needs of the Graphics Language/Interface (GL/I) implementation.

4.10. LPOS to LPOS Communication Services

Service's Interfaces: OSOSI, SAPI

These services support a standardized way of passing information between LPOSs of NGCR operating system(s). It is to be determined whether LPOSs within an NGCR operating system implementation can be supplied by different vendors and plugged into the NGCR operating system. If this is the case, services must support a common set of protocols so that the LPOSs of the system are able to coordinate services and resources with each other. This coordination may be done all at the application level with various application programs communicating with each other through their individual LPOS. This would be fully supported by the SAPI. Alternately, the LPOSs could also communicate and coordinate resource usage directly with each other via the SRAX. This would provide the primitives for such services as automatic dynamic load leveling and automatic reconfigura-

tion. In either case the support of LPOS to LPOS communication will require a common set of OS level messages and protocols whether it is provided by the application, by agreement of the vendors or as part of the NGCR OSS.

4.11. User-Machine Interface (UMI) Services

Service's Interfaces: UMI, SAPI, PS

These services allow I/O to be interchanged between the system and the user of the embedded system through the SAPI or PSEI in an efficient and standardized way. Services that may be included are menu services, windowing services, command line services, parsing services, and pointer device services. These services will interface with low level device services as needed, while presenting a higher level view to the human user. Higher level interfaces for much of this set of services may be provided by the Graphics Language/Interface Working Group (GL/IWG) of NGCR rather than the OSSWG. Note that the User-Machine Interface Services provide the building blocks (menu utilities, command parsers etc.) for building the user interface while the system operator services make available system status and control functions to appropriate application programs with the proper security level.

4.12. Networks and Communications

Service's Interfaces: BAPI, LANI, LHWI, OSOSI, SAPI

These services involve the information exchange between the local processor nodes of an NGCR system.

4.12.1. Network Control and Status

These services provide authorized users the capabilities to determine the status of network components and to control network working parameters. Many of these services logically reside in the LPOS. Other control and status services, however, logically reside in the SRAX.

These services include system startup configuration, network restart, network initialization, network security, network scheduling network monitor, network configuration management, and time. They provide services that allow the network to efficiently use its resources. These services may make use of the OS scheduling services described in the scheduling services section , section 5.17.

4.12.2. Inter-Process Communication

This service allows a local processor node's local operating system to request a procedure, function or transaction to be performed on another processor node or logical resource. There are various forms of inter-process communication, some of which specify the receiver, some specify the sender, some are synchronous (i.e., delay the sender until the communication is completed), some are asynchronous etc. The particular forms that need to be specified by the NGCR OSS are to be determined.

4.12.3. Distributed Voting

This service allows the application to request and collect 'votes' or answers from applications distributed across some communication medium. The request may require processing and/or information from the voters, and the answers returned by the voters may be simple (yes/no) or complex. The resulting votes will be analyzed by either distributed voting services, other services (such as reliability services) or application program(s). Distributed voting services will handle situations where a vote is not received in the appropriate time. This service will likely make use of the synchronization and time services.

4.12.4. Remote Resource Allocation

This service allows a local node to allocate for usage a resource which is physically located at another node within the set of cooperating local processor operating systems or within the set of processors controlled by an SRAX level operating system.

4.12.5. Naming

These services allow the usage of system resources through logical names rather than the actual hardware device naming conventions. Furthermore, they allow the resources of other processor nodes to be accessed via a logical name so that no knowledge of the resource's location is needed and the resource's location may change over time. Logical names are also used by security services to hide resources from unauthorized processes by only letting authorized processes know the logical name that is needed to use the physical resource.

The logical name to physical name relationship can be one to many, many to one, or many to many. Many times one physical resource may have multiple logical names as well as one logical name representing a 'bank' of available physical resources. These services must provide the proper resolution of names, logical and physical, in all of these cases.

4.13. Process Management Services

Service's Interfaces: ARTEI, DBKI, OSOSI, PSEI, SAPI

Typically the following process management services are required by application programs:

a) Create a process and make it ready for execution
b) Destroy a process and recover its resources
c) Evaluate a reference to a process
d) Evaluate a connection to a process, where a connection is logical communication path between any two processes.

4.14. Project Support Environment Services

Service's Interfaces: PSEI

During the system development process there is a need for the Project Support Environment (PSE) to communicate with the system under development. The operating system in the target will need to support that communication. These services may not be available at the Application Program Interface (API) but may be accessed via a different interface. These services may also be removed from the system when it is deployed. The types of

services included here are down-loading of compiled programs and data into the target system, uploading to the PSE of program results and trace information and the interactive debugging by a developer on the PSE of an application running on the target system.

4.15. Reliability, Adaptability and Maintainability Services

Service's Interfaces: BAPI, DBKI, LDI, LANI, LHWI, OSOSI, SAPI

Robustness of a system or application is many times a desirable feature. The services supporting robustness (reliability, adaptability and maintainability) are often implied services in that there is not a direct interface to these services through the SAPI layer of the operating system. Reliability and adaptability services deal with the need for the system to perform functions that the application requests in a timely manner, whenever possible. Reliability is the ability to correctly perform a job to completion, adaptability is the ability to change the system's logical makeup (or jobs to do) over time, while maintainability is the ability to keep the system in operating condition. A highly adaptable system can facilitate the reliability of application's functions.

4.15.1. Fault Tolerance Services

These services allow the system to react to the loss or incorrect operation of system components at various levels of abstraction (hardware, logical, services etc.). The classical model of fault tolerance has a three step approach. The three steps are fault detection, fault isolation, and fault recovery. Typically implementations divide these steps into substeps or integrate them into one or two steps. Additionally, fault diagnosis services support the other steps in the treatment of a fault.

Various fault tolerance strategies, such as checkpointing and voting, are implemented as a collection of services comprising one or more of the steps in the fault tolerance classical model. For example, services involved in implementing a 3 node voting scheme will include a vote comparator service (fault detection), vote analyzer service (fault isolation/fault diagnosis), a service to pass the majority 'answer' through (fault recovery) as well as a service to disable the faulty resource and reconfigure the voters (fault recovery/reconfiguration).

Service categories 'fault tolerance' and 'event and error management' may share services with each other.

4.15.1.1. Fault Detection

Fault detection services are concerned with determining when a fault has occurred in the system. Fault detection services are both passive and active. Active services are those which attempt to determine the status of various system components by testing those components. Passive services, on the other hand, try to ascertain system components by passively gathering information and watching the behavior of the system.

4.15.1.2. Fault Isolation

Fault isolation services attempt to determine the component at fault and segregate the faulty component from the rest of the system. Services may be shared between the fault detection and isolation service library in that they perform both functions.

4.15.1.3. Fault Recovery

Fault recovery services attempt to bring the system into a consistent state. These services may be very interrelated to the scheduling services, network services and data base services depending on the recovery scheme used.

Redundancy of resources is many times needed to support fault recovery. Resources may be data, process, processor, disk drive etc.

As parts of the system fail, it may no longer be possible to satisfy all the requirements of the application. Services to support graceful degradation may be used to ensure that critical activities do not fail.

4.15.1.4. Fault Diagnosis

These services deal with the system's ability to analyze the attributes of a system fault.and determine it's cause. These services tend to be very interrelated with fault detection and fault isolation services.

4.15.2. Fault Avoidance

These services involve the avoidance of faults before a failure in the system component occurs. If a system can detect that the operation of a component is approaching the edge of its operational range then a standby or backup component could be phased in to replace it. Another form of fault avoidance is logging of shocks, temperature extremes etc. so that it can be predicted that a component will not meet its expected service life.

4.15.3. Software Safety

These services involve the system's ability to keep application software from causing harm to the system's software, hardware or user. For instance, a process may attempt to write into another process's memory space without permission.

A good example of a reliability method which may provide software safety is a bounds checker. The checker compares an answer supplied against the bounds. If it is not within the bounds, the bounds checker will not allow the answer to propagate, possibly causing damage to the system's integrity. Additionally, it may send a fault message (or security violation information, depending on the type of answers expected) to the proper service.

To enhance software safety, other services and processes should be only given the resources necessary to complete their job.

4.15.4. Status of System Components

These services involve the obtrusive and non-obtrusive diagnosis of the state of system components. For further explanation of these services see fault detection and fault diagnosis services. These services may additionally need to record and/or display information concerning performance, configuration, and general system information.

4.15.5. Reconfiguration

These services allow the system to reconfigure its view of the world. This services allow the system to substitute different resources to perform system functions such as substitut-

ing a new physical I/O channel to support a logical channel. These services are part of the API but their use may be restricted to specially authorized programs such as those used by the system operator.

4.15.6. Maintainability

Maintainability services provide support for the maintenance of the embedded system. A major component of that support is the collection and logging of information about the operation of the system. Typical information to be logged are:

- Software and hardware errors during operation
- Processes which failed or almost failed to meet scheduled deadlines
- Performance metrics for system tuning
- Times when the system operated in extreme environmental conditions
- Errors reported during startup self-testing
- Attempts to violate rules of the systems security policy

4.16. Resource Management Services

Service's Interfaces: ARTEI, DBKI, LANI, LDI, LHWI, OSOSI, PSEI, SAPI

These services are involved in the management of the systems resources. Resources include CPU, memory, I/O and other physical devices. Services which manage the usage of the CPU are described in Process Management Services, section 5.13.

4.16.1. Memory Management Services

These services support the usage of the LPOS main memory(s). These services supply a virtual view of the memory or memories on the computer as seen by applications and perform the proper mapping of virtual to physical memory (performing any swapping of memory paging needed in the process). Memory management services provide storage to allow process and data migration as well as initialization. The memory manager many times receives requests for service from the process management services to allocate and deallocate memory for process usage. The major services of memory management fall into five categories: allocating physical memory; mapping of logical address to physical storage; memory sharing; extending memory (virtual storage); and protecting user information.

4.16.2. Device Management Services

These services attempt to remove the dependencies on physical resources. The service user sends information to and from the devices by way of logical data structures and/or device service requests. These services mainly serve to supply four functions: device allocation, device control, device status, and device access.

4.17. Scheduling Services

Service's Interfaces: ARTEI, LANI, LDI, LHWI, OSOSI, PSEI, SAPI

These services schedule or arbitrate the usage of various resources of the NGCR OS, particularly the CPU. The scheduling services must be able to queue up requests to use a particular resource. This situation is made more complicated by the common need to schedule processes to run cyclically at a fixed period. When the resources become idle, the scheduler must select one of the 'requestors' of the resources to grant use the resource. These services are listed separately rather than under the services that use scheduling to emphasize

that there should be uniformity and consistency of scheduling across the range of resources.

Typically there are at least two types of scheduling occurring in an operating system: short-term and long-term. Long-term schedulers determine which possible requestors at a given time may actually request a resource. The short-term scheduler selects from among the active 'requestors' which currently have need of the resource and allocates the selected 'requestor' to the resource. For example, if the requestors are processes and the resource is the CPU, then the long term scheduler manages the movement of processes from inactive (waiting in batch queues or in hibernation) to active (in wait or execute). The short-term scheduler, on the other hand, would determine which process should execute next on the CPU. Hybrid services between the two may also be available in the operating system.

When a request for a resource is submitted to the operating system (at some local operating system node), it is not always serviced at that local node. The most advantageous way to service the request may result in part or all of the work being performed at a different processor node. Several reasons may cause this to occur including load balancing, resource availability, computation speedup, hardware preference, and software preference. These services may hide from the application the fact that the functionality was being performed at a different node. This has the advantage that the code needs to know little about the system on which it is running. Alternately, the services may allow the user to specify directly on which logical resource the function should be executed.

The priority scheduling of resources allows the requestor to have associated with it its importance to use the service. More complex schemes also have a criticalness of the request which is used for graceful degradation purposes. The scheduler(s) will use the priority information to arbitrate resource requests and to queue requests in the specific order. A priority scheduler may need to support multi-level queues to support proper execution.

Preemptive schedulers will deallocate a resource from a requestor when certain events occur. Usually this is when a requestor of a higher priority requests the resource or a specified time limit for the resource has expired.

4.18. Synchronization Services

Service's Interfaces: ARTEI, LDI, LHWI, SAPI, LANI, OSOSI, DBKI

These services are involved in the ability to synchronize the operations of other services, functions, processes and/or resources. Services such as distributed voting and remote resource allocation will need to use these services in order to accomplish their required functionality. Synchronization services are needed for both the local processor operating system's operation and the control of the distributed system. Synchronization services may need to use system monitoring services in order to adjust to system changes.

4.19. System Initialization and Reinitialization Services

Service's Interfaces: DBKI, GRKI, LANI, LDI, LHWI, OSOSI, PSEI, SAPI

System initialization includes a complete restarting of the software, starting up the attached hardware subsystems devices, doing subsystem and system self tests and completely initializing the database.

System reinitialization includes restarting the software while using the existing database information. The software may have to be reloaded and the database may have been reestablished by a system recovery. Attached hardware subsystems devices may be reinitialized.

Reinitialization should include a function to restart applications redistributed to other processors after a processor module failure. Within a processor, there should be a function to initialize applications in a system with the existing software but with the database reinitialized. Also within a processor, there should be a function to restart the applications in a system with the existing software and database retained.

4.20. System Operator Services

Service's Interfaces: SAPI, PSE

The system operator needs to access and control the operating system in order to allow the system to perform properly. If a system has an operator the major functions that need to be supported are system control, reconfiguration and status reporting. This service today is usually implemented through a command language interpreter which is an application program that provides access to these services. Note that the User-Machine Interface Services provide the building blocks (menu utilities, command parsers etc.) for building the user interface while the system operator services make available system status and control functions to appropriate application programs with the proper security level.

4.21. Time Services

Service's Interfaces: ARTEI, LHWI, OSOSI, SAPI

The following time management services are likely to be needed:

a) Local Time of Day which includes the time based upon a 24 hour or 12 hour clock.

b) Measurement of elapsed time.

c) Distributed Time which would be a capability to coordinate Local Time of Day maintained by any LPOS.

d) Requests that a Process be delayed for a specified elapsed time.

e) Requests that a Process be delayed until a specific time.

f) Requests for process notification at a specific time or after a specified delay.

5. Target Domains

The domains in which the NGCR OSS are expected to be used vary greatly and vary in several different ways. This section discusses important ways that the target systems differ and the major implications these differences may have on an operating system and on the NGCR Operating Systems Standards. It is expected that there will be a need to tailor or subset the Operating Systems Standards for particular target systems. One concept is that the OSS may not be one standard but a family of standards, each member engineered for a particular class of target systems but having much in common with other members of the

family of standards. Another concept for tailoring is that there will be only one complete set of interfaces implemented with a small OS kernel and a compilation system that links in with each application only the functions or services needed by that application. A third tailoring method could be to expect the OS vendors to provide an OS tailoring tool so that a general OS can be tailored by the system engineer to fit the needs of a particular target system.

Note that there are multiple dimensions along which OS needs vary. This implies that there is no simple way to create a family of OS standards which meets the needs of the different target domains and is also small in the number of member standards.

5.1. Target Processor Interconnection

The processor interconnection, i.e., the means of communication between processors, can vary from a single processor system with no connection to other processors to an LPOS system connected via a full network to many other processors of various types.

5.1.1. Single Processor Systems

In a single processor system the OS has little if any need to support network communication services except as an alternate API for Inter-Process Communication (IPC). If there is more than one program on the processor then some form of inter-process communication will often be needed to allow the application programs to coordinate with each other.

5.1.2. Multiprocessor Systems

A multiprocessor system is one where multiple processors share common memory. The sharing of common memory allows the implementation of fast synchronization as well as fast communication between processors. Many multiprocessor systems have local (or private) memory in addition to the shared memory. This can increase the efficiency of execution of programs, but having to support two varieties of memory can add to the complexity of the OS.

5.1.3. Distributed Systems

A distributed system is one which has multiple processors without any shared memory. There are three kinds of distribution but real systems are often complex combinations of the three kinds of distributed and multiprocessor systems. The kinds of distribution are classified according to the interconnection mechanism:

- Backplane Interconnection
- Local Area Network Interconnection
- Full Network Interconnection

5.1.3.1. Backplane Interconnection

Processors which are interconnected by a high speed backplane have a fast communication mechanism, but depend upon the capabilities of the backplane hardware for synchronization and communication. The operating systems in a backplane interconnected system need to use more complex synchronization techniques which tend to be somewhat slower than in multiprocessor systems, however the SAPI for those services may not differ.

5.1.3.2. LAN Interconnection

In an LAN interconnected system the processor to processor communication is via the local area network. This kind of distribution is limited to processors that are on the same local network without any storage of messages between nodes of the network. In this type of system the reliable communication needed for inter-process communication (IPC) can be achieved via simple send and acknowledge schemes although the communication speed adds significant overhead time compared to backplane interconnection.

5.1.3.3. Full Network Interconnection

In a full network interconnected system there are multiple LANs or at least multiple LAN segments with bridges and/or gateways which use store and forward communication schemes. The lack of a direct network connection between processors significantly increases the complexity and overhead required to produce a reliable IPC. Depending upon the organization of the SAPI the difference in complexity may not be visible to the application designer except as longer communication delays of a particular implementation.

5.2. Security

A major goal of the NGCR program is to provide systems which can be used to meet the security needs of the Navy. The security needs of a project vary with the project and can be met in a variety of ways. For this model we will group the security needs according to the type of access control required.

5.2.1. Targets with No Security Requirements

Many Navy systems have no special security requirements because they either do not process classified or sensitive material or because they operate in facilities that meet the security requirements with physical security alone. For these systems the OS would probably not implement the security services. For compatibility with secure systems, however, there may be limits on the functionality provided by the OS through the API.

5.2.2. Targets with Discretionary Access Control Requirements

Many non-embedded computer systems provide some form of Discretionary Access Control (DAC) , usually via passwords and file permissions. The Department of Defense - Trusted Computer System Evaluation Criteria defines the classes of computer security or trustedness provided by computer systems. Class C systems provide discretionary access control which is a means of restricting access to objects based on their identity or the identity of the groups to which they belong. Supporting discretionary access controls requires that there be interfaces to the OS by which the permissions, passwords etc. can be changed. Also required is a means of reporting or processing access violations.

5.2.3. Targets with Mandatory Access Control Requirements

Mandatory Access Control (MAC) is a means of restricting access to objects based on the sensitivity of the information contained in the objects and the formal authorization of subjects to access information of such sensitivity. MAC adds only minor interface requirements but can significantly increase the amount of implicit services required of the OS.

5.3. Robustness

Robustness refers to how well the system can be expected to continue operation and how quickly it can be repaired when some part malfunctions, it also refers to features of a system that prevent unsafe actions from taking place.

5.3.1. Reliability and Availability

Reliability and availability can be achieved by fault avoidance and/or fault tolerance. Fault avoidance is achieved by increasing the reliability of the hardware components and applying conservative design practices. Fault avoidance is primarily achieved via hardware design. Fault tolerance is achieved by the use of redundancy and requires software support.

Fault tolerance is concerned with data integrity and processing integrity. Data integrity deals with providing survival of data when components fail and is closely linked with database system technology (data replication, atomic transactions etc.). Processing integrity tries to insure correct and continuous processing across instances of component failure. Processing integrity is especially important in safety-critical real-time systems.

5.3.2. Software Safety

Software safety has the goal of preventing any unsafe action even in the face of incorrect software processing. Software safety often involves independent monitors (either software or hardware) which check the "reasonableness" of results or actions of the system.

5.3.3. Maintainability

Maintainability of a system describes how quickly and correctly system errors or failures can be determined and corrected. The OS can provide services that aid in maintainability including the logging of system errors (hardware and software), reporting of built-in-test results and usage logs for scheduling preventative maintenance.

5.4. Richness of the Set of OS Services

The target domains of the NGCR OSS vary in the number of OS functions needed. A small, single purpose system or one with an extreme emphasis on performance may need a very small, finely tuned OS; a general purpose system may need many services to be provided by the OS. This is somewhat a matter of philosophy. One view is that the NGCR OSS should be a minimal set of interfaces, leaving to the particular project the job of implementing higher level functionality in a project specific manner. Another view holds that the NGCR OSS should be a full set of interfaces so that there can be a large amount of software portability among projects.

5.5. Real-Time Requirements

Real-time computing is dead-line driven; i.e., computing that involves intricately intertwined computation deadlines (often imposed by external stimuli) on short time scales. While the NGCR OSS are for embedded systems, not all embedded systems have real-time requirements and those target systems with real-time requirements have various needs.

5.5.1. Non-Real-Time Target Systems

Some embedded systems are used for applications that have no stringent time demands. Examples of these systems are systems used for planning and for maintenance support.

These systems still need an efficient high performance operating system but the results' correctness do not depend upon their being available at a particular time.

5.5.2. Real-Time Target Systems

Real-time systems are probably the most common targets for the NGCR OSS compliant operating systems. Therefore the OSS must surely support the needs of embedded real-time systems. Common OS real-time requirements include the ability to specify a dead-line for completion of a process, the ability to specify that a process is to be run cyclically with a specific period, the ability to specify that one process or program is more important to the system than another and the ability to rearrange the importance of processes or programs as the operating mode of the system changes.

5.5.3. Critical-Time Target Systems

Critical-time targets are those which have real-time response requirements that, if not met, result in system failure. These systems may even require that results not arrive too early, but precisely when it is needed. Examples of these target systems are many safety critical systems such as flight control systems. To support these target systems the OSS will need to allow flexible and predictable scheduling.

Appendix A

A. NGCR OSSWG Background

The U.S. Navy has embarked on a new computing resources standardization effort called Next Generation Computer Resources (NGCR). This program is designed to fulfill the Navy's need for standard computing resources while allowing it to take advantage of commercial products and investments and to field new technology advances more quickly and effectively. The program revolves around the selection of standards in 10 interface areas. One of these is an operating system standard. The general requirements for this operating system are that it be Ada-oriented, real-time, distributed/networked, multi-level secure, reliable and realizable on heterogeneous processors. The effort to establish such an interface standard was initiated at the start of 1989 and will draw on industry expertise. An initial operating system interface standard is expected in 1993 and the final standard is expected to be usable in the procurement of Navy systems in 1995.

The Navy has a long history of developing and using standard computer products. When computer technology was in its infancy, the Navy wielded significant influence in the market, setting its own requirements and developing its own computer designs, including Instruction Set Architectures (ISAs). Standard computer implementations (i.e., buying "boxes") and upward compatible ISAs have been the foundation of the Navy's computer policy. This policy has been motivated by the fact that software can adapt a common computer design to meet many different applications

But the Navy's current computer standardization approach is having difficulty remaining competitive in an environment where rapidly changing technologies permit more efficient and effective solutions to the range of Navy computing system requirements.

Thus the objective of the NGCR program is to restructure the Navy's approach to acquisition of standard computing resources so as to take better advantage of commercial advances and investments. It is expected that this new approach will result in reduced production costs (through larger quantity buys), reduced operation and maintenance costs, avoidance of replication of Navy RDT&E costs (for separate projects to develop similar computers), and more effective system integration.

The proposed new approach is an open systems approach based on the establishment of standards in 10 interface areas:

- Multisystem Interconnects:
  - Local Area Network - SAFENET I
  - Local Area Network - SAFENET II
  - High Performance Local Area Network
- Multiprocessor Interconnects:
  - Initial Backplane
  - High Performance Backplane
  - Switch Network
- Operating System
- Data Base Management System
- Programming Support Environment
- Graphics Language/Interface

Application of these interface standards will change the Navy's approach from one of buying standard computers to one of procuring computing resources which satisfy the inter-

faces defined by the standards. These standards will be applied at the project level rather than a Navy-wide procurement level.

These interface standards will be based, to the greatest extent possible, on existing industry standards. In cases where existing industry standards do not meet Navy mission-critical needs, the approach is to further enhance the existing standards jointly with industry, thus assuring the most widely-accepted set of commercially-based interface standards possible.

The NGCR Operational Requirements describe some of the desired characteristics of the computer systems which can be procured using the new interface standards:

- a full-range family of computing resources, related through a set of interface standards, in a wide range of performance levels; software compatibility at appropriate levels is a necessary part of the "family" relationship
- integration of multiple, dissimilar (heterogeneous) processors
- internal and external standard interconnection; i.e., an internal computer interconnection (bus) to provide for growth in internal capability by configuring more modules and an external interface to provide for combining computing systems
- incremental computing system growth; i.e., if a new function is needed, new modules or computers would be added to a system, and adding the new components would not require replacing the old system.

An operating system interface standard is a key element in the success of NGCR. In this work the OS is that set of functions which control operation of all the computing system hardware and software elements of a platform in a coordinated, uniform manner that is consistent with the mission of the platform. The OS provides functions for system and platform management and control The OS functions include system initialization, fault tolerance and recovery, global resource allocation and inter-process communication. The OS will have components in each processing element. The OS interface standard is not a design of the OS component of each processing system but is, in part, a specification of an application program interface common to all computing elements. (The appropriate specification level for this interface must be determined). This provides the basis for system-wide dynamic task and resource allocation. Global dynamic task and resource allocation is the basis for system-wide fault tolerance and recovery in heterogeneous processing systems. Some implementations of the OS Standards will provide the ability to achieve multi-level security at the system level. Conformance to other Navy directives requires that the OS be Ada-oriented.

The above system characteristics and OS requirements have some implications for the OS interface standard. Most important of these is that the OS "standard" will most likely actually be a family of compatible interface standards, although the exact nature of the family relationship is yet to be determined.

Appendix B

B. Acronyms

API	Application Program Interface
ARTE	Ada Run Time Environment
ARTEI	Ada Run-Time Environment Interface
BAPI	Binary Application Program Interface
CPU	Central Processing Unit
DAC	Discretionary Access Control
DBKI	Data Base Kernel Interface
DBMS	Data Base Management System
GL/I	Graphics Language/Interface
GL/IWG	Graphics Language/Interface Working Group
GRKI	Graphics Kernel Interface
I/O	Input / Output
IPC	Inter-Process Communications
ISA	Instruction Set Architecture
ISO	International Standards Organization
LAN	Local Area Network
LANI	Local Area Network Interface
LDI	Local Device Interface
LHWI	Local Hardware Interface
LPOS	Local Processor Operating System
MAC	Mandatory Access Control
NGCR	Next Generation Computer Resources
OS	Operating Systems
OSI	Open Systems Interconnect
OSOSI	LPOS to LPOS Interface
OSS	Operating Systems Standards
NGCR	Operating Systems Standards Working Group
PSE	project support environment
PSEI	Project Support Environment Interface
ROM	Read Only Memory
RPC	Remote Procedure Call
RT	Real Time
RTNI	Real-Time Non-Intrusive Testing
SAPI	Source Application Program Interface
SAPI	Source Application Program Interface
SRAX - C	SRAX - Centralized Part
SRAX - L	SRAX - Local Part
SRAX	System Resource Allocation Executive
TCB	Trusted Computing Base
UMI	User Machine Interface

Index

NGCR

(Next Generation Computer Resources)

OSSWG

(Operating Systems Standards Working Group)

AVAILABLE TECHNOLOGY REPORT

Version 1.3

09/14/1990

Table of Contents

Section 1

INTRODUCTION

The objective of the Next Generation Computer Resources (NGCR) Program is to standardize Navy mission critical computer interfaces and computer component interfaces. With these standardized interfaces, industry will be better able to provide computing resources that meet Navy needs. The interface standards are to be widely available (i.e., non-proprietary) and, if possible, widely utilized within industry.

The NGCR Operating Systems Standards (OSS) is one of the sets of standards which is essential to the timely and cost effective acquisition of the majority of the next generation of Navy mission critical computing systems. NGCR OSS assists the Navy in efficiently providing a wide range of performance, compatible computing services, and functionality levels.

The primary objective of the NGCR Operating Systems Standards Working Group (OSSWG) will be the selection from commercial standards and, where these standards are not available or are not adequate, the development in conjunction with industry of a set of interface standards for a family of distributed target operating systems to cover the complete spectrum of Navy combatant and other mission critical use.

1.1 Scope

The NGCR interface standards, while being incrementally developed, are to be sufficiently in place so that the Navy can begin acquiring systems utilizing those standards by 1996. Prototype systems using the OSS are to be developed with a contract award scheduled for September of 1990.

The period of NGCR Operating Systems Standards development began in FY89 and continues through FY95. The initial OSS will be available for use in acquisitions starting in FY93.

The initial range of applications includes as many types of computing as possible, from just above the single dedicated processor to as high as can be obtained on networked, heterogeneous, modularized backplane bus architecture computing systems. Networking is to be done using NGCR Local Area Network (LAN) standards and, as appropriate, other MIL-STD links.

1.2 Purpose

The Available Technology Subgroup of the NGCR OSSWG (OSSWG-ATSG) has conducted a survey and analysis of operating systems technologies. The results of the survey and analysis are documented in this NGCR OSSWG Operating Systems Technology Report.

The results documented here will be used to aid the work of the OSSWG Requirements and Approach Subgroups as well as the entire OSSWG in developing the NGCR OSS.

Section 2

BACKGROUND

In this section, basic operating systems concepts and terminology are reviewed.

2.1 Operating Systems

[JENSE81, PETER84, TANEN85]

There are many definitions and approaches for describing operating systems. Peterson and Silberschatz open their book [PETER84] with the following statement:

"An operating system is a program which acts as an interface between a user of a computer and the computer hardware. The purpose of an operating system is to provide an environment in which a user may execute programs. The primary goal of an operating system is thus to make the computer system convenient to use. A secondary goal is to use the computer hardware in an efficient way."

Traditional operating systems have been built with the above incentive. However, the development of microcomputers and distributed architectures changed both the emphasis and the order of significance of the goals as listed above. Tanenbaum and Van Renesse [TANEN85] express it as follows:

"An operating system is a program that controls the resources of a computer and provides its users with an interface or virtual machine that is more convenient to use than the bare machine."

2.2 Distributed Systems

[DAVIES81, SLOMA87, CHAMB84]

A multi-programmed system is one that provides the interleaved execution of two or more programs on a single processor. In this case, the operating system must coordinate the use of the hardware resources among the running user programs. The coordination of the use of processor time is called scheduling. The coordination of all other resources (memory, file storage, I/O devices, etc.) is called resource allocation. A multi-programmed system must protect each user from other users that are executing during the same time period. In addition, many multi-programmed systems provide support for interprocess communication (IPC) and sharing of data between cooperating processes.

In computer systems consisting of more than one processor and more than one memory unit, the problem of coordinating the use of

these hardware resources is further complicated. This resource allocation / scheduling function can be performed in one process of the system (centralized) or by a set of processes running on a set of processors in the system (distributed). Centralized techniques are simpler and similar to techniques used in single processor systems. For this reason they are often used in tightly-coupled, multi-processor systems. However, the resource allocator / scheduler process itself is a system resource. The single instance of this resource often creates a bottleneck in the system and results in poor system reliability. In addition, this technique requires total system state information to reside in one location. This is often very difficult in loosely-coupled, wide-area networks of processors. It becomes even more complicated when one tries to define what a distributed operating system is. Jensen [JENSE81] considers global management to be the key concept:

"A distributed operating system is one which provides the same sort of global resource management that a centralized (uniprocessor or multiprocessor) OS does, but without depending on the existence of physically or logically centralized resources such as shared primary memory or global system state. Thus, a distributed OS differs from a network OS in its objective of providing global resource management (such as multi-node co-scheduling) for one or more distributed applications rather than resource sharing (such as network file services and remote procedure calling) among separate applications. And a distributed OS differs from a centralized (uniprocessor or multiprocessor) OS in its approach of not depending on the existence of physically or logically centralized resources. Consequently, a distributed OS is able to provide traditional system-wide OS resource management services even though the processors have disjoint primary memories and are physically dispersed and loosely coupled by a communication subnetwork. Note that global resource management involves more than providing the abstraction of a single machine through network transparency; it requires that resources be managed as an actual single machine in the best interests of the entire system."

2.3 Real-Time Systems

[AGRAW89, STANK88]

In the context of this report, real-time systems, and critical time systems, also referred to as hard real-time systems in the literature, are defined as those systems in which the functional correctness of the system depends not only on the correctness of the logical results of computations performed by the system, but also on the times at which the results are produced. Mission critical real-time, or critical-time, systems are characterized by the fact that severe consequences will result if logical as well as timing correctness properties of the system are not satisfied. Typically, such a system consists of a controlling

system and a controlled system. Thus, the controlled system can be viewed as the environment with which the computer interacts.

Most of the hard real-time computer systems are special-purpose and complex, require a high degree of fault-tolerance, and are typically embedded in a larger system. Also, real-time systems have substantial amounts of knowledge concerning the characteristics of the application and the environment built into the system. A majority of today's systems assume that much of this knowledge is available a priori and, hence, are based on static designs. The static nature of many of these systems contributes to their high cost and inflexibility. Future generation hard real-time systems should be designed to be dynamic and flexible, but this requires different approaches to analysis and design of not just the OS but also the applications.

Timing constraints for tasks can be arbitrarily complicated, but the most common timing constraints for tasks are either periodic or aperiodic. An aperiodic task has a deadline by which it must finish or start, or it may have a constraint on both start and finish times.

An NGCR real-time system must be able to meet the timing requirements of a variety of periodic and aperiodic requests. Research and system building experience have shown that speed alone does not adequately solve the problem of these requirements. The mechanisms for supporting real-time scheduling of system resources requires the integration of the hardware and the operating system subcomponents in much more deterministic and predictable manners.

Ideally, the computer should execute critical-time tasks so that each task will meet its timeliness requirement, whereas it should execute the non-critical tasks in accordance with application-specified policy (e.g., the average response time of these tasks is minimized). The need to meet the requirements of individual critical-time tasks is one issue that makes the problem of designing a real-time system a difficult problem.

Low-level application tasks, such as those that process information obtained from sensors or those that activate elements in the environment, typically have stringent timing constraints dictated by the physical characteristics of the environment. A majority of sensor processing is periodic in nature. Some of these periodic tasks may exist from the point of system initialization, while others may come into existence dynamically.

Consequences of not meeting timing constraints are usually applications dependent. For embedded Navy operational systems, catastrophic results may occur if critical-time task deadlines are missed. Resources needed for highly critical tasks in such systems generally have to be preallocated so that the tasks can execute without delay. Other non-critical-time or "soft" tasks may also

have time constraints associated with them, but missing those deadlines may have much less severe effects. In most systems, there is a complicated mix of critical-time, soft real-time, and non-real-time tasks.

While traditional real-time systems have been focused on low-level sampled data subsystem applications (such as pipelined signal processing and sensor/actuator feedback control), the requirement for meeting stringent real-time constraints is expanding into larger, more complex, more distributed systems, such as C3I and surface/subsurface combat platform management. Such applications are intrinsically more dynamic and stochastic in their behavior and thus violate the premises which underlie almost all conventional real-time operating systems [JENSE88]. Another important distinction between real-time and non-real-time operating systems is the balance of performance optimization between normal and exception cases. Non-real-time systems, both software and hardware (cf the RISC CPU approach), optimize performance of the normal case and are willing to pay a significant penalty in exception cases. But real-time systems are often required to perform their best in exception cases such as hostile attack or failures, even if that means some of the normal, most frequent cases must suffer higher overhead [JENSE88].

In summary, mission critical real-time systems differ from traditional systems in that deadlines or other explicit timing constraints are attached to tasks; the systems are in a position to make compromises; and faults, including timing faults, may cause catastrophic consequences. This implies that, unlike many non-real-time systems where there is a separation between correctness and performance, for real-time systems, these factors are very tightly interrelated. Thus real-time systems solve the problem of missing deadlines in ways specific to the requirements of the target application.

2.4 Real-time Distributed Systems

Many of the Navy's applications areas, including the C3 arena, are naturally distributed and complex; therefore, future systems must integrate distributed resources. Not only should the target operating system provide communication mechanisms, but it should use them in such a way as to unify this distributed set of system resources. All this must be done while still considering the real-time requirements of the system.

Section 3

AVAILABLE TECHNOLOGY SURVEY AND SYNOPSIS

3.1 Introduction

In this section, fielded operating systems and areas of operating systems research of interest to the NGCR OSS Program are surveyed. The survey is by no means complete.

3.2 Operating Systems Synopses

3.2.1 ACCENT

Accent is the operating system designed for Spice, a large network of personal workstations at the Computer Science Department of Carnegie-Mellon University. Begun in 1980, this system is intended to support the research community of CMU as a general purpose distributed LAN. Accent is a communication oriented, object based operating system providing a multi-process user view. Each process has a disjoint address space and communicates with other processes via "ports". Communications are node transparent, thus servers are also node transparent. Accent provides all user resources via interprocess communication ports. Even internode communication and virtual storage are performed via user-level processes, so any multilateral control, decentralized or otherwise, is processed at the user level. [RASHI81] [BALL82] [LOCKE84]

3.2.2 Alpha

Alpha is an OS for the mission-critical integration and operation of large, complex, distributed real-time systems (e.g., DoD C3I,combat platform and battle management). Alpha is a global OS, and explicitly manages all resources directly with actual application-specified task completion time constraints. Its performance is optimized for important exception cases rather than the most frequent cases. Alpha is object-oriented, and includes real-time distributed data management mechanisms (for maintaining problem-specific consistency and correctness constraints) integrated into its kernel. Alpha arose from the Archons Project at CMU, and was funded by many DoD and industry organizations. A prototype has been operational there and at General Dynamics Corp. since Fall 1987. The focus of the Archons Project has moved to Concurrent Computer Corp. in Boston MA, where it continues to be sponsored in part by DoD, and includes a second-generation Alpha design and implementation. Alpha is portable, non-proprietary, and in the public domain for U.S. Government use. The initial ports are on 68030-based and MIPS-based multiprocessors; these multiprocessors are networked with the NGCR standard XTP real-time transport protocol running on FDDI. Alpha can co-exist with UNIX

on any node, and a POSIX-compliant version is planned. Additional versions of Alpha being developed jointly by Concurrent and its partners include a multilevel secure one, and a compatible subset for traditional low-level sampled data subsystems. Pilot versions of Alpha will be installed at various Government and industry contractor facilities in Fall 1990. Point of contact: E. Douglas Jensen, Concurrent Computer Corporation, Technology Way, Westford, MA 01886, (508)-392-2999

3.2.3 ALS/N

ALS/N (Ada Language System / Navy) is the Ada compiler being developed for the UYK43. The compiler will be tested by a number of Navy labs including NOSC and NSWC in FY89.

3.2.4 AMOEBA

Amoeba is a distributed operating system being developed at the Vrije Universiteit in Amsterdam, The Netherlands. This system is composed of a number of processors in a pool, none of which are dedicated to a single user. This is in contrast to, for example, Accent, which is a set of distributed personal workstations, each of which is dedicated to a single user, but which can use resources from other connected workstations and hosts.

Amoeba views the system as a collection of processes with disjoint address spaces and ports, using 3.2.message communications for all interprocess communications.

At the operating system level, very little decentralized resource management takes place. Resources such as disk, files, and names, are encapsulated by processes operating at the user level. No team decisions are made at the operating system level, since each node acts unilaterally. [TANEN81] [LOCKE84]

3.2.5 ARGUS

3.2.6 ARTS

Advance Real-Time Technology (ART) is an open project interested in the development of advanced real-time technologies. The project is funded by ONR and presently includes efforts by CMU, SEI and IBM-FSD. The objective of the ART project is to develop theoretical foundations, distributed real-time system technology and programming language support that will facilitate the development of distributed real-time systems with understandable, predictable, and maintainable behavior. They are interested in the development of real-time scheduling theory, distributed real-time operating systems and distributed real-time databases. The project is presently developing a real-time DOS called ARTS and an experimental system called Real-Time Mach. Point of contact: Hide Tokuda, Computer Science Department, Carnegie-Mellon University, Pittsburgh, PA, 15213, (412) 268-7672.

3.2.7 ARTX

The Ada Real-Time Executive (ARTX) is a real time operating system from Ready Systems (Palo Alto, CA). ARTX implements the full range of Ada Semantic operations, including the complete Ada tasking model. It has many of the features of their VRTX system. Ready Systems have RTAda-MP, which supports tightly-coupled multiprocessor systems (680x0 based). It adds Ada multiprocessing capabilities to ARTX.

3.2.8 ASOS

3.2.9 ATES 43

ATES 43 is designed to operate in a complex of UYK 43 multiprocessor computers. It is an evolution from ATES, an operating system in use on the early CG 47 Class ships. ATES 43 will be used on the later CG 47 Class ships and on the DDG 51 Class ships. It provides a uniform fault tolerant message passing protocol among processes whether in the same or different computers. It will recover automatically from all single point hardware and software faults. It provides for preemptive priority based scheduling within a computer and limited priority based scheduling among processes in different computers.

3.2.10 BiiN OS

BiiN OS was the operating system of the BiiN family of multiprocessor computers. BiiN/OS was designed to operate in a distributed computing environment; multiple, geographically distributed BiiN computers could be unified into a single "distributed" system through the BiiN/OS. BiiN/OS addressed fault tolerance, real-time, and security concerns. With respect to fault tolerance, it dynamically configured hardware modules to provide three levels of fault tolerance, which represented different tradeoffs between fault tolerance and performance. With respect to real-time, it provided traditional real-time application support, including preemptive, priority-based scheduling over a set of multiprocessors. With respect to security, it provided discretionary access control through a combination of capabilities and access control lists. In addition, BiiN/OS offered a UNIX-compatible operating system environment through its BiiN Open Standard Interface Extension (BOSIX) tools. Points of contact: Steve Tolopka, Andy Crump, BiiN, 2111 N.E. 25th Avenue, Hillsboro, Oregon, 97124-5961, (1-800) 252-2446. The BiiN project is no longer active.

3.2.11 BSO Real-time Craft OS

Boston Systems Office (BSO), Waltham, MA developed the BSO/Realtime Craft Operating System. The operating system is packaged in a PROM containing executable code. Interface libraries

are available for programming in C, Pascal, Ada and other higher level languages. The response speed of the system does not degrade as the number of tasks increases, and an unlimited number of tasks can be handled. Versions are available for use with 8086, 68000 and 32000 families of processors. [Falk p66]

3.2.12 CAIS-A

See CAIS-A in section 4.

3.2.13 CCIU-OS

In 1981, CECOM (the U. S. Army Communications-Electronics Command) established the Command and Control Information Utility (CCIU) program at the Jet Propulsion Laboratory (JPL). The program was oriented toward a survivable distributed information processing system architecture to organize automated tactical Command and Control resources.

A demonstration system which implemented most of the major survivability features, but was of reduced functionality was written in Modula 2 within a VMS shell and runs on a network of VAX processors. [DAVIS87]

3.2.14 C-Executive

C-Executive real-time operating system is from JMI Software Consultants, Spring House, PA. It is written in C and provides a C-language environment. Data-move routines in C-Executive are optimized for byte-oriented operation. About 95 percent of the C source code in which C-Executive is written remains the same regardless of the processor used with C-Executive. It runs on 14 different processors, and will soon be available for some reduced-instruction-set computer processors. Functions performed by assembly-language code in C-Executive include context switching, task scheduling and interrupt handling. Device drivers are written in C. [Falk p58]

3.2.15 CHAOS/GEM

The Generalized Executive for Multiprocessors (GEM) operating system (as used in Ohio State University's Adaptive Suspension Vehicle) uses a process-mailbox program model and supports multiple models of communication which may be chosen so as to best implement the inter-subsystem communication semantics of the application domain. The Concurrent Hierarchical Adaptable Object System (CHAOS) extends GEM to support the notion of objects interacting via invocations. Objects can be of different weights and a variety of invocation primitives are provided.

The GEM kernel was developed originally for Intel 8086/MultibusI-based multiprocessors and has been used on a number of robotics applications. It has recently been upgraded to operate on Intel 80386/MultibusII-based multiprocessors, taking advantage

of protected virtual memory, MultibusII message passing, and 80386 debugging features.

CHAOS was developed as a library executing on the GEM kernel. It has now been ported to run on the MACH system using the C-Threads library. Versions of this port are available for a Sun workstation and an Encore MMAX.

[1989 Workshop on Operating Systems for Mission Critical Computing, September 19-21, 1989.]

3.2.16 Choices

The Choices family of operating systems is designed as a toolkit for building efficient parallel, distributed, real-time, embedded and high-performance operating systems. It exploits class hierarchies and object-oriented programming to facilitate the construction of customized operating systems for both shared memory and networked multiprocessors. The system is entirely written in C++ except for a few lines of assembler. [Campbell, R.H., J.H. Hine, and V.F. Russo, University of Illinois at Urbana-Champaign, "Choices for Mission Critical Computing", from the 1989 Workshop on Operating Systems for Mission Critical Computing, September 19-21, 1989.]

3.2.17 CLOUD

Clouds is a distributed operating system being developed at the Georgia Institute of Technology. It has received major funding from NSF, NASA, and RADC. Originally, the primary design goal of Clouds was the support of reliable, fault-tolerant distributed computing. The object/thread programming model (in which the traditional "process" is decomposed into an object, which serves as an abstraction of storage, and a thread, which serves as an abstraction of computation) was conceived as a means to an end, the end being reliable, fault-tolerant distributed computing. However, it has become an end in itself; the support and exploitation of the object/thread programming model is now the overriding theme of the Clouds research. Research topics include operating system support for objects, replication and consistency management using objects in a distributed environment, and programming language/methodology/tools support for programming distributed applications using objects. Points of contact: Rich LeBlanc and Partha Dasgupta, School of Information and Computer Science, Georgia Institute of Technology, Atlanta, GA, 30332, rich@gatech.edu, partha@gatech.edu.

3.2.18 CMP/OS

A Common Module Processor (CMP) and Operating System have been developed at Hughes Aircraft Company. Some features include real-time, mission critical avionics, combined signal and data processing, and Ada support written in Ada. [Miyahara, G.K. and Cynthia L. Allyn, Hughes Aircraft Company, "Realtime Operating

System for Secure, Mission Critical Avionics Systems" from the 1989 Workshop on Operating Systems for Mission Critical Computing, September 19-21, 1989.]

3.2.19 CRONUS

Cronus is a distributed operating system being developed by BBN and is funded by RADC, NOSC and ESD. It incorporates many desirable DOS features such as heterogeneity, transparency and object oriented programming as well as high level features such as survivability and replication mechanisms, multi-cluster and database access and distributed monitoring and control facilities. Cronus is presently being used by several Navy projects such as Fleet Command and Control Battle Management Program (FCCBMP). It is also being used as a basis for study at NOSC of Navy DOS requirements. Point of contact: Andres Echenique, BBN, 10 Moulton Street, Cambridge, MA, 02238, (617) 873-4304.

3.2.20 CRYSTAL

Charlotte is the operating system being constructed as part of the Crystal project, intended to provide a facility for performing research into distributed applications at the University of Wisconsin. Crystal is an outgrowth of the earlier Arachne and Roscoe projects at Wisconsin.

Charlotte's model of computation is that of a hierarchically arranged set of processes passing messages within a statically defined partition of the entire network. The partition is set up manually by a host, and the kernel limits interprocess communication to processes within each partition. It is through the partitioning that multiple "jobs" are executed.

Process management is performed by multilateral management of a set of process controllers (squad of processes). Each member of the process management controller set maintains its version of the state of the set of nodes which it controls, as well as information on other node sets. [FINKEL83] [LOCKE84]

3.2.21 CXOS

3.2.22 DARK

Distributed Ada Real-Time Kernel (DARK) was developed by The Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA. It was tested on a distributed 68020 target, integrated with Inertial Navigation System, and Beta tested at several acceptor sites. [1989 Workshop on Operating Systems for Mission Critical Computing, September 19-21, 1989.]

3.2.23 DINOS

The Distributed Network Operating System (DINOS) is the operating system of a network of distributed processors called the

Siemens Local Network (SIELOCnet) implemented as a research effort by Siemens AG, Munich, Germany. The network is connected by an optical bus providing high speed communications.

DINOS is designed to be a decentralized operating system for controlling the SIELOCnet system. The model of computation for DINOS is a hierarchically structured set of user processes. A group of processes with a common address space is the distributed unit, while a group of distributed units performing a single job is an execution unit. A distribution unit is allocated to a single processor. [FRIED83] [SCHMID82] [SCHMID83] [LOCKE84]

3.2.24 DRAGON/MELODY

DRAGON SLAYER is a distributed operating system with an adaptive file system called MELODY. They were designed for realizing a distributed real-time system working in a hazardous environment. Future work with this system will be in a distributed testbed environment which is at Wayne State University, Detroit, MI. [Wedde, H.F., Ghasem S. Alijani, Dorota Baran, Gookhai Kang, Bo-Kyung Kim, "DRAGON SLAYER/MELODY: Distributed Operating Support for Mission Critical Computing" from the 1989 Workshop on Operating Systems for Mission Critical Computing, September 19-21, 1989.]

3.2.25 E10.S

The 10.S system is a distributed telephone exchange switching processor. The primary design goal is high reliability in the presence of one or two processor failures. The system architecture consists of a set of task forces (logical machines) which share a common address space, reside on a single node, and contain one or more processes. These task forces are centrally allocated to the available nodes. Individual processes communicate by messages and, in the case of processes within a particular task force, potentially by shared memory. [GATEF81] [MAISON81] [LOCKE84]

3.2.26 EDEN

Eden is a project of the University of Washington, Seattle, Washington to build and use a distributed computing environment. Its primary research interest is in the user interface to the distributed environment and to facilitate research into distributed computing. [LOCKE84]

3.2.27 ELXSI

EMBOS is the operating system for the ELXSI System 6400 multiprocessor developed by ELXSI, Santa Clara, California. Since this system is actually implemented as a shared memory multiprocessor (with up to fourteen CPUs), it should perhaps not be described here, but its operating system specifically prohibits the use of any interprocess shared memory, either within or above the operating system level. [OLSON83] [LOCKE84]

3.2.28 FlexOS

FlexOS is a modular real-time operating system developed by Digital Research (Monterey, CA). Modular operating systems let users select only those functions they need. Stripped of unnecessary modules, the operating system then places less demand on computer resources. The FlexOS operating system on each computer has a kernel and a memory manager, but the use of other FlexOS modules is optional. [Falk p64]

3.2.29 43RSS

See 43RSS in section 4.

3.2.30 GALAXIE

GALAXIE is a decentralized system designed by the Centre National d'Etudes des Telecommunications, Lannion Cedex, France, to provide real-time control for the operation of a telephone switching exchange. This is a decentralized system prototype consisting of a set of 8080 and Z80 processors and suitable communications links.

The operating system uses decentralized control to handle process loading, process-processor binding , topological maps, hardware configuration, and processor naming. Loading and process-processor binding decisions are made with the assistance of the applications programmer who can override the automatic decisions, or who provides software to assist in making those decisions. [ANDRE82] [LOCKE84]

3.2.31 GUARDIAN (Tandem)

3.2.32 GUARDIAN (Honeywell)

Guardian is a real-time, multi-microprocessor operating system designed for decentralized control of embedded systems. It was developed by Honeywell for an architecture consisting of six microcomputer processing elements and twelve shared memories fully interconnected by a crossbar switch.

This was basically a research effort, investigating the possibility of implementing a secure version of MULTICS. It was primarily a paper system which never actually went into production. Work on GUARDIAN was terminated in the early 1970s. (Carl Reinert)

3.2.33 HARMONY

Harmony originally came out of the Canadian National Research Council and is now developed and marketed by DY-4 Systems (Nepean,

Ontario). Harmony is a real-time operating system that was originally designed to be a multiprocessing kernel. A copy of the kernel resides in each processor. Although other kernels such as MTOS and VRTX use the same arrangement, Harmony data structures aren't replicated. The kernel for each processor has its own data structures, and there's typically little sharing of data between processors. Since global memory is used less, contention for the multiprocessing bus - a problem that plagues many multiprocessing kernels - is reduced. This system does not offer any security-relevant features. Point of Contact: Jeremy James, Manager of Software Technology, DY-4 Systems Inc. (613) 596-9911.

3.2.34 HARTOS

HARTOS is the distributed real-time operating system for HARTS (Hexagonal Architecture for Real-Time Systems). It is being developed at The University of Michigan. [1989 Workshop on Operating Systems for Mission Critical Computing, September 19-21, 1989.]

3.2.35 HERBERT-II

HERBERT-II by Arizona State University and Codex Corporation, Tempe, AR, is a modification to Codex ISOS, a Unix-like operating system. It has a distributed file system and a simple distributed database management system. Herbert-II has used the International Standards Organization (ISO) defined protocols (i.e., physical, datalink, network, transport, session, presentation, application). It has been implemented on three smart terminals manufactured by Codex.
[MILLE83]

3.2.36 HOPS

Honeywell CSDD has been doing research and development on software fault tolerance and distributed systems. They have implemented a runtime for executing fault tolerant distributed applications and language and translator support for their development. The system is called HOPS (Honeywell Object oriented Programming System). A subset of a typical BM/C3 application has been implemented using the language support to evaluate the fault tolerance and distributed system mechanisms. Point of contact: Jon Silverman hi-csc!wilbur!silver@umn-cs

3.2.37 HP-UX

This product was developed by Hewlett-Packard, of Fort Collins, CO. HP-UX is organized as a hybrid real-time Unix. Hewlett-Packard inserted kernel pre-emption points throughout the kernel code. It allows pre-emption of an in-process kernel service in order to schedule a higher priority real-time program in response to an interrupt. Additional real-time features were

implemented as extensions. [Rauch-Hindin]

3.2.38 HXDP

The Honeywell Experimental Distributed Processor (HXDP) was developed as a research effort into real-time distributed processing by the Honeywell Systems and Research Center, Minneapolis, Minnesota. This was an early investigation into the issues of distributed computers in a real-time environment.

The HDPX executive computational model consists of a set of processes and messages. Processes (Virtual Processing Elements) are active entities which receive messages when they are ready, sending messages as needed. The executive does not concern itself with ensuring the reliable transmission, receipt, or sequencing of individual messages, other than ensuring that each message, if delivered, is individually unchanged during transmission. [BOEBERT78A] [BOEBERT78B] [CORNHILL79] [JENSE78] [LOCKE84]

3.2.39 IDRIS

IDRIS is a large operating system used for both general-purpose and real-time computing. It was developed by Whitesmiths, Westford, MA. IDRIS supports the Posix standardization effort. [Falk p60]

3.2.40 iRMX

See iRMX in section 4.

3.2.41 ISIS

The ISIS system is an extension of a conventional distributed operating system to support fault tolerance. It insulates the application programmers from the details of fault tolerant programming. This system is currently running at Cornell University and current work is aimed at integrating the ISIS more fully into the operating system. The project is supported by DARPA and NSF [BIRMAN85].

3.2.42 KSOS

The design of the Kernelized Secure Operating (KSOS) was developed by the Ford Aerospace and Communications Corporation (FACC) and Logicon and funded by NSA, DARPA, and the Navy [LANDW83]. This system was verified and rated at the A1 level. The KSOS project is no longer active.

3.2.43 LOCUS

LOCUS is a distributed UNIX operating system developed at UCLA for use by the UCLA computer science research community. The primary goal was to build a system with multiple processors running

UNIX, with a replicated location transparent file system.

The computational model for LOCUS is exactly that of UNIX. Programs written for a uniprocessor VAX UNIX can generally execute without change on LOCUS. [POPEK81] [WALKE83] [LOCKE84]

3.2.44 MACH

Mach is a multiprocessor-oriented operating system for a distributed environment being developed at CMU and is funded by DARPA. It approaches issues involved with multiprocessors, heterogeneity, transparency, and object oriented programming. Mach is presently being used by several projects at CMU and being used and extended by a number of corporations, universities and research laboratories. NOSC has plans to add Mach to its DOS testbed in FY88-89. Point of contact: Rich Rashid, Computer Science Department, Carnegie-Mellon University, Pittsburgh, PA, 15213, (412) 268-2627.

3.2.44.1 RTMach

The goal for Real-Time Mach (RTMach) is to migrate the ARTS scheduler and scheduling tools to Mach. Also modifying Mach's virtual memory management to support locking threads in memory. Point of contact: Hide Tokuda, Computer Science Department, Carnegie-Mellon University, Pittsburgh, PA, 15213, (412) 268-7672.

3.2.44.2 Trusted Mach

The Trusted Mach project is a DARPA-sponsored research effort of Trusted Information Systems, Inc. The goal is to build a version of Mach - Trusted Mach - that meets the B3 level of protection as specified in the National Computer Security Center (NCSC) Trusted Computer System Evaluation Criteria (TCSEC), the so-called "Orange Book" [TCSEC 85]. The project adopts the idea of "incremental reference monitors." At the lowest level is the Trusted Mach Kernel. At the intermediate level is the reference monitor composed of the kernel and a trusted name server. At the highest level is the reference monitor composed of the kernel, a trusted name server, and other trusted servers. Thus far, work has concentrated on the kernel level of a single machine. At this time, the Trusted Mach project is utilizing a Spring 1988 version of Mach. Since this version is not kernelized, the effort cannot yield a trusted operating system. The unkernelized version of Mach is serving as a platform for research into multilevel security, not as a base upon which to build a trusted system. The development of a trusted version is tied to the completion of Mach kernelization. Points of contact: Steve Walker, Marty Branstad, Trusted Information Systems, Inc., 3060 Washington Road (Route 97), Glenwood, MD, 21738.

3.2.45 Maruti

Maruti is a real-time distributed operating system being

developed at the University of Maryland. It has been funded, in part, by the U.S. Army Strategic Defense Command and the Office of Naval Research. MARUTI focuses on real-time and fault-tolerance requirements. It supports both in an object-oriented framework. With respect to real-time requirements, MARUTI supports guaranteed service scheduling. That is, once it accepts a task, MARUTI guarantees that the timing constraints of the task will be met. It utilizes replication and consistency control mechanisms to implement user-specified fault-tolerance constraints. Point of contact: Ashok Agrawala, Department of Computer Science, University of Maryland, College Park, MD, 20742, (301) 454-4968, agrawala@mimsy.umd.edu.

3.2.46 MEDUSA

The Medusa operating system was developed by Carnegie-Mellon University. It is a distributed operating system for Cm* multimicroprocessor architecture. It is a message passing operating system which has distributed, disjoint utilities. The Medusa operating system is divided into disjoint utilities. Utilities are distributed among the various available processors, with no guarantee that any particular processor contains a copy of the code for any particular utilities. Since no processor is guaranteed to be capable of executing any particular piece of code, it may be necessary for a program's flow of control to switch processors when it invokes a utility function. In Medusa, messages provide a simple mechanism for cross-processor function invocation. The invocation message contains parameters for the function invocation, as well as an indication of a return pipe; the return pipe is analogous to a return address for a subroutine call. It does provide some insight into architectures where certain function (utilities) may not be available on all processor but may exist on only special purpose processors (a symbolics processor, a signal processor).

3.2.47 MFM

The Message Flow Modulator (MFM) system is based on a model of message filtering. It was developed at the University of Texas using Gypsy and the Gypsy tools. Both the design and the code have been verified. In spite of this state-of-the-art verification, the MFM system is only rated at the C2 level. This is because the message filtering model does not include the multi-level and compartmented labeling facilities required for the B level. This work was funded and is currently being used by the Navy [LANDW83].

3.2.48 MIKE

MIKE is the operating system of the Distributed Double-Loop Computer Network (DDLCN) being developed at the Ohio State University, Columbus, Ohio, to investigate fundamental problems in distributed processing and local networking. MIKE is a meta-

operating system (i.e. it is built on top of the existing host operating systems) providing the appearance of a distributed system to applications software running on the hosts. The hardware architecture is constructed as part of OSU's research into communications media, and is not tailored for operating system use other than for communications. [LIU82] [TSAY81] [LOCKE84]

3.2.49 MIMAS

MIMAS is a network operating system currently under design at the University of Strathclyde, Glasgow, Scotland. This system is being planned for the purpose of investigating the structure and operation of a decentralized operating system on a local area network. [BLAIR82] [LOCKE84]

3.2.50 MOSI

Microprocessor Operating Systems Interfaces (MOSI) was prepared by Working Group 855 of the Microprocessor Standards Committee, sponsored by the IEEE Computer Society. This interface standard (IEEE Std 855) designates the capabilities required by various microprocessor-based applications. MOSI is no longer active.

3.2.51 MTOS

The multiprocessor, multitasking operating system (MTOS) was developed by Industrial Programming, Inc. It is a family of available real-time operating systems which include MTOS-UX/386 (a fully dynamic system supporting complete intertask coordination, synchronization and resource management) and MTOS-UX/Ada (a dynamic operating system that can handle multiprocessing for a real-time executive and the Ada language. Applications can be written in a high-level language or in assembler to request any one of the MTOS services.

3.2.52 MUNET

MuNet is a distributed processing system developed at the Massachusetts Institute of Technology, Cambridge, Massachusetts. The system ha. been designed to study distributed processing with particular emphasis on the construction of an extensible (and contractible) computer with the property that increases in processing capability can be made linearly with respect to cost. This extensibility is achieved by using topologies in which each node is connected only to a bounded set of neighbors via point-to-point inte .aces. [HALSTEAD80] [LOCKE84]

3.2.53 ORKID

3.2.54 OS-9

OS-9 was developed by Microwave Systems (Des Moines, IA) OS-9 runs on 680x0 and maybe other microprocessors. It is a real-time operating system that offers high degree of configurability. Users can easily customize I/O file management and drivers. It is written in assembly language. [Falk p65]

3.2.55 Pave Pillar

The Ada Avionics Real-Time Software (AARTS) Operating System (AOS) is designed for the PAVE PILLAR avionics technology. The AOS is used to provide control for and system services to the Mission Data Processors. The Mission Data Processors will be one or more VHSIC 1750A modules. There may be multiple mission processors. The modules communicate via redundant inter-module communications buses (PI-Bus). The AOS is a straight forward operating system that provides system services to the program application(s) and special control functions. Additionally, the AOS has a fault-tolerant capability provided by redundant resources and a distributed control strategy.

3.2.56 PHOENIX

The PHOENIX Project, at the University of Virginia, is researching the problems associated with (1) developing a high-performance UNIX for embedded applications with real-time response requirements, (2) modifying an operating system and applications remotely without halting the computer, and (3) recovering after power failure or crash. Currently a high-performance version of UNIX, written in Modula, is being used to evaluate techniques that address these problems. This project is supported by the Army Research Office [COOK85a]. The PHOENIX, also UNIX-based, focuses on needs that are important to the platform systems, (i.e., real-time response updates, and recovery for embedded applications).

3.2.57 POSIX

The IEEE Portable Operating System (POSIX) interface standard is based on earlier UNIX operating system interfaces. This standard defines an application program interface to an underlying set of operating system functions; it does not specify the structure, functions, or performance of the underlying operating system beyond the specific functionality visible at the application program interface level.

This standards committee, IEEE P1003, has created an initial version of the interface standard which completed balloting in August, 1988, and is therefore identified as IEEE 1003.1-1988. In addition, this committee includes a number of smaller working groups which are expected to present extensions or application-specific interfaces for this standard for official balloting before 6/90; draft versions of these extensions are

currently available. The working groups include 1003.2 (Shells and System Utilities), 1003.3 (Test Method Specifications), 1003.4 (Real-Time Extensions), 1003.5 (Ada Bindings), 1003.6 (Security), 1003.7 (System Administration), 1003.8 (Networking Services) and 1003.9 (Fortran Language Binding).

3.2.58 PSOS (SCG)

Probable Operating System(PSOS) was developed by Software Components Group, Santa Clara, CA. It features Real-time kernel which is used as the kernel base for several other real-time operating systems such as Regulus and VxWorks. Has file and debug options that can be added. [Falk p57-58]

3.2.59 PSOS (Honeywell)

The Probably Secure Operating System (PSOS) is a capability-based system. The design was formally specified and verified but an implementation was not initiated. The design was developed by FACC and Honeywell and was funded by NSA [LANDW83].

3.2.60 Regulus

Regulus was developed by Alcyon, San Diego, CA It features a real-time version of Unix compatible with Unix System V. It uses pSOS kernel in combination with its own version of Unix. It runs on 80386, 680x0, and 32000. Real-time enhancements to Regulus include multitasking priority based scheduler, memory areas that can be directly accessed and shared by tasks, and intertask event signaling. [FALK p58]

3.2.61 RIG

The Rochester Intelligent Gateway (RIG) is designed as a "front end" system to provide a uniform interface between a user terminal and any of a variety of mainframe time-sharing systems. Designed about 16 years ago, it is in use as a heterogeneous distributed system providing for research into distributed algorithms, computer vision analysis, and design automation for VLSI. RIG was intended to provide an extensible, flexible "glue" to construct a coherent time-sharing system for a research community. [BALL76] [LANTZ82] [LOCKE84]

3.2.62 RMS68K

RMS68K was a product was developed by Motorola Semiconductor Products, Inc for the 68K family of processors.

3.2.63 RSS/M

See RSS/M in section 4.

3.2.64 RTU

3.2.65 SCOMP

The Honeywell Secure Communications Processor (SCOMP) received an A1 rating. This system used a hardware box called the SPM, security protection module, to monitor transfers on the bus without CPU interference resulting in faster performance than KSOS achieves. This work was funded by Honeywell, NSA, DARPA, Defense Communications Agency, and the Navy. The system is currently used by several government agencies [FRAIM83].

3.2.66 SDEX/44

See SDEX/44 in section 4.

3.2.67 SDOS

The Secure Distributed Operating System (SDOS) is in experimental development at Odyssey Research Associates, Inc (ORA). The system is being designed and built to meet TCSEC B3 security and assurance requirements. SDOS borrows many of its concepts from Cronus, such as the basic object-oriented client-server model. However, the system architecture has been redesigned to provide multi-level security, enhanced identification and discretionary access control, configuration security, audit, COMSEC protection and TCSEC assurance. [Varadarajan, R., J.R. McEnerney, and D.G. Weber, Odyssey Research Associates, "The Secure Distributed Operating System - An Overview" from 1989 Workshop on Operating Systems for Mission Critical Computing, September 19-21, 1989.]

3.2.68 SDX

Standard Distributed Executive (SDX) is a real time executive which operates in a multicomputer environment via the Shipboard Processing and Display System (SHINPADS; NOTE: SHINPADS is a registered trademark of the Canadian Department of National Defence) Serial Data Bus (SDB) network. SDX supports the United States Navy and Canadian Navy standard 16-bit general purpose computers: the AN/UYK-20, AN/UYK-20A, AN/UYK-502, AN/UYK-505, and AN/UYK-44 computers. SDX was developed by Unisys, St. Paul, MN.

SDX was based on SDEX/44 (a U.S. Navy standard executive for 16-bit computers) and supports real time priority scheduling as well as message handling capabilities. SHINPADS SDB systems implement hardware and software redundancy and reconfigurability to provide fault tolerance.

The SDX based Distributed Operating System (DOS) provides real time error detection, error isolation, hardware reconfiguration, and software reconfiguration capabilities. The System Loader component of DOS supports the loading of configurations or individual modules into remote processors via the SDB.

For more information see SDX in section 4.

3.2.69 SHOSHIN

Shoshin is a distributed testbed computer developed by the University of Waterloo, Waterloo, Ontario, Canada. Shoshin was developed to serve as a testbed by researchers in distributed software engineering and distributed algorithms. [TOKUDA83] [LAU83] [LOCKE84]

3.2.70 SIRIUS-DELTA

The Delta real-time transaction based operating system has been developed under the auspices of the French Ministry of Industry and the Institut National de Recherche d'Informatique et d'Automatique (INRIA), France. This research project has as its primary goal the design of a data base system, and the Delta operating system is designed to support a real-time distributed data base. [GLORIEUX81] [LELANN81] [SEDILLOT80] [LOCKE84]

3.2.71 Spring Kernel

The Spring project at the University of Massachusetts is conducting research into next generation hard real-time systems. The project has four major thrusts: the development of dynamic, distributed, on-line real-time scheduling algorithms; the development of the Spring Kernel, a distributed real-time system; the development of multiprocessor nodes to support the kernel and the scheduling algorithm; and the development of real-time tools. Point of contact: John Stankovic, University of Massachusetts, Dept. of Computer and Information Science, Amherst, Ma, 01003, (413) 545-0720.

3.2.72 SPRITE

Sprite is an experimental network operating system under development at the University of California at Berkeley and is funded by DARPA. Motivation for the new operating system came from trends toward networks, large memories and multiprocessors. Sprite is part of a large project called SPUR whose goal is to develop a high-performance multiprocessor workstation with special hardware support for LISP. Sprite focuses on issues involved with transparent network file systems, large variable-size file caches, shared address spaces, and process migration in a distributed environment. Point of contact: John Ousterhout, Computer Science Division, University of California, Berkeley, CA, 94720, (415) 642-0865.

3.2.73 StarLite

StarLite is a research project that is exploring new ideas for operating system structuring, interface design, analysis, and

implementation. The prototyping environment, which executes on Sun workstations, supports the development and execution of software for uni- or multi-processors, as well as distributed systems. StarLite provides a standard UNIX interface together with an implementation strategy that addresses the critical system needs of high-performance, openness, and predictability. [Cook, R.P., Department of Computer Science, University of Virginia, "The StarLite Operating System" from the 1989 Workshop on Operating Systems for Mission Critical Computing, September 19-21, 1989.]

3.2.74 STAROS

StarOS is an operating system developed for the Cm* computer at Carnegie-Mellon University. Cm* is a large asymmetric shared memory multiprocessor developed in a research project into distributed processing, containing 50 DEC LSI-11 microproccessors. StarOS was developed to provide an object-based testbed to allow experimental user software to take advantage of the Cm* design. [JONES79] [TRIGG81] [LOCKE84]

3.2.75 TCNA

The Tightly Coupled Network for VHSIC Architecture operating system has been developed by the Westinghouse Electric Corporation for the RADC. It is designed to run on the Westinghouse Electric Corporation's Tightly Coupled Network for VHSIC architecture. The operating system consists of four parts: (1) Local Operating System (LOS) - which resides in each of the processing modules. It provides the task management, task scheduling, I/O interfaces, debugging capabilities, and fault-tolerance facilities; (2) Cluster Operating System (COS) - which resides in multiple general purpose computers in each cluster of the TCNA. It makes decisions that effect the entire cluster; (3) Regional Operating System (ROS) - which resides in multiple general purpose computer modules within each region. It controls placing modules off-line due to failure; and (4) Group Operating System (GOS) - which reside in three identical general purpose processors, all of which receive the same information and vote on what action to take. Point of contact: TBS.

3.2.76 TRON

Tron is an operating system standard that was originally initiated by Ken Sakamura, University of Tokyo. It has gained wide support in Japan including companies such as NTT, Fijitsu, Mitsubishi, and Toshiba. TRON designates a family of architectures, operating system kernels, and VLSI CPU chips. It is an open specification intended to foster the development of compatible products by many vendors. There are several versions of the TRON architecture; ITRON, for embedded industrial systems, BTRON, for business oriented workstations, CTRON (central TRON), for large fileservers in a networking environment and MTRON (macro

TRON), for interconnecting "intelligent objects". [SAKAM87]

3.2.77 V DISTRIBUTED SYSTEM

The V Distributed System is a network operating system being developed at Stanford University and is funded by DARPA, the National Science Foundation and AT&T Information Systems. It is used to explore issues in distributed systems and focuses on areas such as high performance interprocess communication including multicast, process migration and the distributed scheduling of programs. There is also research being conducted in the areas of replication, distributed atomic transaction management and multiprocessors. Point of contact: David Cheriton, Stanford University, Computer Science Department, Stanford, CA, 94305-2140, (415) 723-1054.

3.2.78 VersaDOS

VersaDOS was an operating system developed by Motorola Semiconductor Products, Inc. for early microprocessors (8-bit) and later, 68K family of processors.

3.2.79 VRTX

VRTX is manufactured by Ready Systems (Palo Alto, CA). Chief among VRTX's facilities is multitasking. The task is to real-time programming what the procedure is to more conventional programming; a structural unit that can be considered separately from other units. VRTX has been designed to maximize its performance for high-priority tasks, unlike non-real-time systems, which are designed to be "fair" and allocate an equal overhead to all tasks. -VRTX is also a compact system, occupying less than 6K bytes of code space. Most importantly, VRTX is deterministic. Its behavior can be completely predicted in all circumstances.

3.2.80 VXWORKS

The VxWorks operating system from Wind River Systems (Emeryville, CA) is designed to debug and run real-time tasks developed on a Unix system. Target computers can use 68000, 68010, or 68020 processors, and can run alone or networked with other computers that run VxWorks or Berkeley 4.2 Unix over Ethernet or a backplane bus. VxWorks uses calls very similar to Unix calls.

3.2.81 WIS0S

The WIS Operating System, funded by the Joint Project Management Office, is designed to aid in the modernization of the World Wide Military Command and Control System (WWMCCS). It provides a minimal kernel to optimize local area network IPC and provides fast context switches. It provides clearly delineated address spaces, basic mandatory access control, and communication

control to help support security in the kernel. Outside the kernel, security is supported by "alias" processes and an authentication agent. Its multi-level, modular design is suited for evolutionary change. The status of a prototype, written in Ada, is unknown at this point. Information is needed. The WIS Operating System also addresses fault-tolerant, secure, real-time programming in a distributed environment and is designed specifically for battle management support.

3.2.82 XTS-200

3.2.83 ZMOB-OS

ZMOB is a "mob" of 256 processors developed at the University of Maryland at College Park, Maryland. The operating system for ZMOB is called MOBIX, and has been designed to facilitate artificial intelligence and general computer science research [Trigg 81]. The operating system is designed to allow ZMOB to act as an extension of the host, making normal forked UNIX processes run in a truly parallel fashion by shipping them to individual Z80 processors for execution. [BANE81] [LOCKE84]

Section 4

DETAILED OPERATING SYSTEMS SURVEY

4.1 Introduction

After collecting the brief product descriptions as in section 3, the next step was to collect more detailed technical information. This information was collected and organized based on the 16 service classes as described in the NGCR OSSWG Reference Model Document.

4.2 Preliminary Operating Systems Survey Summary Results

4.2.1 Alpha Survey Summary

Alpha is intended for an unconventional but vital and rapidly growing segment of the real-time field: the mission-critical integration and operation of large, complex, distributed real-time systems, such as: DoD C3I, battle management, and combat platform management; industrial factory and plant automation; and even commercial online transaction processing.

Integration and operation brings coherence to physically separated computing nodes, and involves providing:

* a user interface for distributed application programming;
* system-wide (i.e., trans-node) resource management for functionality such as control and coordination (i.e., execution correctness and data consistency), and fault recovery, in support of distributed application programs;
* interoperability with other systems;
* man/machine interfacing. (OSSWG refers to such an OS as an "SRAX".)

Thus, Alpha is optimized for (but not limited to) large, complex, distributed real-time applications characterized by the uncertainty which pervades warfare (and many other endeavors): predominately aperiodic tasks; dynamic and stochastic behavior; demand for resources which usually exceeds the supply; and requiring run-time resolution of resource dependencies and conflicts. Despite these attributes (which are diametrically opposed to those of traditional real-time low-level sampled data subsystems), the application tasks nonetheless have mission-critical time constraints, both hard and soft.

To accommodate such environments, Alpha departs from the traditional real-time approach of statically defining exactly what will happen only under a necessarily limited number of anticipated operational conditions. Instead, Alpha strives to be exceptionally dependable with respect to effectiveness, survivability, and safety, by dynamically adapting so as to do the best (as defined by the users) that it can under the current resource and mission conditions--e.g., ensuring that as many as possible of the most important of time constraints are met, in accordance with

user-specified policies. To this end, Alpha manages all physical and logical resources directly with application-defined actual task completion time constraints (deadlines are a simple special case) and relative importances, instead of using artifactual priorities. Alpha is able to behave as deterministically as the application actually requires and is willing to pay for (e.g., in excess assets and reduced flexibility), and is able to present deterministic abstractions for the users if desired; but it seeks to accomplish these ends as much as possible with non-deterministic means in order to improve robustness and cost-effectiveness (analogous to the use of non-determinstic routing in long distance telephone and wide area data networks).

Alpha's design and implementation also reflect the nature of its application domain in that its performance is optimized for the most important exception cases, such as arise in emergencies due to hostile attack or faults, rather than for the most frequent (normal) cases as is ubiquitous practice in non-real-time systems (cf. the RISC philosophy).

The approaches to incorporating a global (system-wide) OS in a system can be placed into three major classes:

1] it may be an application on the local OS's--e.g., BBN's Cronus and Apollo's NCS;

2] it may co-exist with the constituent subsystem local OS's (e.g., UNIX, ARTX) on partitioned processors of their nodal multiprocessor hardware;

3] it may be native on its own system integration and operation hardware nodes, which then physically and logically connect to the subsystem local nodes and OS's.

The first of these does not provide the global OS adequate authority to meet Alpha's coherent computing responsibilities. Alpha can readily accommodate the second case--e.g., Release 3 will co-exist on multiprocessor nodes with Concurrent's real-time UNIX and JMI's C Executive initially. But Alpha is intended more for the third case because of both technical reasons-- e.g., superior performance can be achieved--and logistical ones--e.g., minimized impact on the many subsystem contractors and on the pre-existing subsystems. Such a configuration would typically include:

* conventional low-level real-time sampled data subsystems, in which centralized applications and local OS's like ARTX or UNIX execute on local hardware nodes, and interface with local i/o devices;
* all integrated into a single coherent computer system by a new system integration and operation subsystem, in which distributed applications and the Alpha distributed OS execute on an additional set of distributed hardware nodes;
* all interconnected with (one or more instances of) SAFENET.

Alpha includes a kernel having an open, specified client interface (among other interfaces), which is subject to potential standardization. Certain non-real-time OS's take the approach of providing a minimal kernel for managing hardware resources, and moving as much OS functionality as possible up to user state programs (sometimes termed "servers"). Alpha shares this

philosophy and its benefits in general, but adapts it to better meet the requirements of the large, complex, distributed real-time application environment. For example, Alpha includes kernel-level mechanisms for distributed atomicity, permanance, and concurrency control; these support its real-time distributed object store and fault tolerance facilities which are clients of the kernel. Alpha carries the principle of policy/mechanism separation to great lengths in order that application-specific cost/performance tradeoffs can be applied to all of its services.

The client programming model presented by Alpha's kernel is an object one. Depending on the OS built on Alpha's kernel, the kernel abstractions can either be confined for use by the remaining layer(s) of the OS which may in turn provide the applications with a different set of abstractions (for example, Concurrent will offer an optional POSIX compliant--and thus less capable--interface), or be passed upward to the applications augmented with additional or modified services (which is what the native Alpha OS does).

Alpha arose as the first systems effort of Jensen's Archons Project on new paradigms for real-time decentralized computer systems, which began in 1979 at CMU. Alpha design was started in 1985 and the prototype ("Release 1") was operational at CMU and at its initial early user General Dynamics/Ft. Worth in the Fall of 1987. Alpha was sponsored at CMU primarily by the USAF Rome Air Development Center (RADC) and the US Naval Ocean Systems Center (NOSC). It was also sponsored in part by DARPA and a number of industrial corporations, including IBM, General Dynamics, and Sun Microsystems.

In response to requests from Alpha sponsors and prospective users that the technology be expeditiously transitioned from research into practice, the focus of Alpha activity moved (to Kendall Square Research briefly and then) in the Fall of 1988 to Concurrent Computer Corp./Boston. It continues to be sponsored in part there by RADC, with additional support from Concurrent.

Concurrent is performing an all new, second generation, commercial quality Alpha design and implementation in a series of increasing functionality releases. The first is Release 2, which is scheduled for pilot installation at a number of major Government and industry facilities in the Fall of 1990, with subsequent releases to follow.

Releases 2 and beyond are portable, non-proprietary, open, and in the public domain for U.S. Government use; licenses for non-Government uses are intended only to control what can be called the "Alpha" OS. Alpha will initially be released on Concurrent's MIPS-based multiprocessors interconnected with FDDI (NGCR's SAFENET II hardware and XTP transport protocol), and will be ported to other manufacturers' hardware.

Both RADC and Concurrent still sponsor Alpha research at CMU, and will add work at MIT beginning in 1990. Concurrent is teamed with several other industrial organizations in various aspects of Alpha design, including:

* a B3 multilevel secure Alpha with SRI International;
* a version of ARTX/VRTX with Ready Systems which is Alpha

compatible-- providing the Alpha programming model (in addition to the traditional VRTX-32 one), so that the same software development techniques and tools can be used for both local centralized application programming and global distributed application programming, and having an "SRAX-to-LPOS" interface with Alpha which facilitates a collection of local ARTX's being integrated into a large, complex, distributed system (this same kind of Alpha compatibility will be offered in a version of Concurrent's real-time UNIX RTU, and is being negotiated with several other vendors of commercial real-time OS's and execs);

* a real-time distributed data architecture system for Alpha with Xerox;

* an expert system-based adaptive fault tolerance strategy subsystem with GE.

A number of DoD's top 10 prime contractors are also actively engaged with Concurrent in transition and experimental evaluation of Alpha in applications such as: ground-based battle management, C3I, and air defense; combat systems on surface and subsurface ships; and avionic mission management. Similarly, civilian industrial factory automation (most notably automobile manufacturing) and commercial financial institutions are considering Alpha. (Because these contractor, industrial, and commercial corporations all expect their potential use of Alpha would provide them competitive advantages, most of them presently require that we not disclose their identities).

4.2.1.1 Operating System Service Classes

4.2.1.1.1 General Requirements

Alpha is a general-purpose real-time distributed OS, optimized to support the entirely different computing paradigm required for large, complex, distribututed real-time applications. It is extremely adaptable so as to be readily configured for a wide range of problem-specific functionality, performance, and cost requirements (including being configureable as a traditional, small scale, low-level uniprocessor/multiprocessor OS). Alpha does not accept the traditional view that its real-time responsibility to the user ends with rapid interrupt response and context switching times; in large, complex real-time applications, the strong correlation between starting a task quickly and it completing on time is not present; thus Alpha continues to provide run-time resolution of dynamically arising resource dependencies and conflicts for all tasks according to their time constraints and importances.

Similarly, Alpha's distribution services are not limited to internode communication and resource sharing via conventional network utilities. Because a distributed real-time system is mission-oriented, it runs distributed applications which require distributed (internode) resource management. Conspicuous among these requirements are distributed task: integrity--continuing computational progress despite errors and partial (i.e., node or

communication) failures; and concurrency control--maintaining correct collective execution and mutually consistent distributed (replicated and partitioned) data, despite asynchronous concurrent execution. Alpha meets these distributed application requirements with unusual real-time distributed data and execution management mechanisms in its kernel and facilities in its system and user layers.

Alpha's native programming model (user interface) is a new object-oriented one especially suitable for writing distributed real- time applications, with threads which reliably and transparently span physical nodes bearing their attributes such as time constraints, robustness, etc., and with block-structured mechanisms for time constraints, transactions, and exception handling. This programming model will also be offered by Concurrent on its real-time UNIX, and and Ready Systems on its ARTX/VRTX (others are being planned).

Alpha is constructed on a small kernel of powerful, general mechanisms which are used in a uniform manner: for example, all physical and logical resources--such as CPU cycles, I/C, memory pages, synchronizers, atomic transactions--are managed according to the same user-specified policy for dealing with task completion time constraints and importances; every type of exception, from time constraint expiration to transaction abort to node failures to hardware checks, is handled with the same block structured construct; and any application can access any resource--code, data items, devices, processors--as though everything were local to it.

Access to the kernel facilities is through a kernel interface library. The initial libraries are for programming in C and C++.

4.2.1.1.2 Architecture Dependent Interfaces

Alpha encapsulates i/o devices, special-purpose hardware units, and external systems as objects; for example, when UNIX is installed as a co-resident OS on Alpha nodes, it appears as an object to Alpha.

4.2.1.1.3 Capability and Security Interfaces

Alpha's lowest level mechanisms for protection are capabilities and separate address spaces for each object instance and thread. Alpha system and application entities can use kernel services to enforce restriction on the propagation of capabilities among protection domains.

Abstractions such as access lists, users, groups, etc., can be implemented by system-level objects, using the capablity mechanisms. Attributes such as groups, users, etc., can be attached to threads and will be automatically propagated by the kernel along with the thread as it moves through the distributed system; thus every activity can have an associated identity for access control purposes.

A B3 multilevel secure version of Alpha is being developed in conjunction with SRI International.

4.2.1.1.4 Data Interchange Interfaces

Because Alpha is homogeneous in the sense that its nodal instances are designed and implemented to the Alpha specification, no data interchange services in the usual sense are necessary within an Alpha system. Data interchange services will be provided at Alpha's application layer for communication with other systems.

4.2.1.1.5 Event and Error Management Interfaces

Alpha employs the same block-structured mechanisms for error management as for transactions and time constraints. An exception handling clause can be associated with each type of error, each instance of an error, each object operation invocation, or any desired combination of these. After Alpha performs its own cleanup in response to errors, it vectors the affected thread to the appropriate application-specified exception handling clause.

4.2.1.1.6 File Interfaces

Alpha is object oriented and thus provides an object store rather than a traditional file system.

Any object may be declared permanent, in the sense that a non-volatile representation of the object's state resides in the object store. Any object may support atomic transaction-controlled updates to its permanent representation, thus providing *for consistent*, failure-atomic typed data storage. The object store is capable of supporting directory objects to map logical names into object capabilities (which are, themselves, system-provided logical names, and are used exclusively, even within the kernel, except within the object store implementation itself).

Should a particular file interface be required, it is possible to create a permanent object class whose interface provides the required file semantics (e.g., operations such as open, close, read, write).

Because Alpha's object store is based on the real-time data management mechanisms in its kernel, all file services are inherently real-time. Alpha also provides for the traditional style of real-time OS secondary storage and file system features, such as pre-allocation of resources, synchronous (non-cached) writes, and contiguous files.

4.2.1.1.7 Generalized I/O Interfaces

All i/o takes place as operation invocations on objects. Alpha's general i/o facility provides for generic i/o operations on device objects.

Release 2 of Alpha includes a simple command interpreter modeled after the UNIX shell. A version of Alpha is planned which includes POSIX compliance.

Release 2 of Alpha can also coexist and communicate with UNIX on the same node, or on different nodes. Non-time-critical man-machine interactions can be supported by the UNIX system.

4.2.1.1.8 Networks and Communications

Alpha is a distributed OS in that its instances not only communicate but also cooperate at the kernel level in order to efficiently accomplish decentralized management of global resources and provide a real-time distributed programming model. This is in contrast to distributed OS's that have essentially uniprocessor kernels, in which even the lowest levels of communication are provided by client-level programs, while the kernel itself simply provides access to the network device. Communication across protection domains in Alpha is provided through object operation invocation. If the invoking object and the invoked object are on different nodes in a distributed system, the invocation is implemented as a reliable RPC, with special enhancements in support of time-driven orphan detection and elimination, distributed exception handling, and global time- driven resource management.

Operation invocation provides location-transparent distributed access to objects and resources (encapsulated by objects) across the system. All resources are named by kernel-supported logical identifiers (capabilities), and the Alpha invocation protocols support network-wide logical addressing using these identifiers. The use of logical naming at the lowest levels of the system allows transparent name mappings that can be one-to-many, many-to- one, or many-to-many, in addition to the usual one-to-one. These mappings are supported at the network level, using logical addressing and the broadcast and multicast capabilities of the underlying communication medium. Each logical identifier in Alpha is globally unique.

Alpha's RPC is built on the NGCR SAFENET XTP real-time transport protocol (Concurrent was contracted by the USN to work with Protocol Engines, Inc. and the SAFENET committee to define XTP's real-time feature.

Below XTP, Alpha includes communication software for the NGCR SAFENET II FDDI and Ethernet LAN networks.

Since Alpha can co-exist in a multiprocessor node with UNIX, it can have access via shared memory to all UNIX network and communication services such as TCP/IP.

4.2.1.1.9 Process Management Interfaces

Alpha is object-oriented, its programming model being based on passive threads. Object instances and classes can be created and deleted dynamically, as can threads.

Alpha's kernel provides primitives for object and thread management, including the above functions and others. References to objects and threads (capabilities) are evaluated when they are invoked.

A single thread executing in a single object instance is a

reasonable approximation of the classical notion of a "process."

All communication above the kernel of Alpha, both among and within its instances, is object operation invocation.

4.2.1.1.10 Project Support Environment Interfaces

Alpha is currently an execution environment OS, although it provides the mechanisms required to support a future native software development. Software development is presently done on UNIX either remotely or locally (when UNIX is installed on one or more Alpha nodes).

Source-level symbolic debugging of the kernel code (written in C++) is currently operational, using a modified version of the GNU gdb debugger's remote debugging facilities. This is a generic low-level debugger interface that is applicable to multiple-language debugging. It is planned that this will be expanded to application-level debugging as system implementation progresses.

4.2.1.1.11 Reliability, Adapatability, and Maintainability Interfaces

Alpha provides kernel-level mechanisms for atomic transactions and replication. Application-specific policies at the system level define consistency, correctness, recovery, etc. for threads and objects, as well as the replication strategy.

Alpha's capability-based location-transparent naming of objects supports the transparent application of a number of higher-level fault tolerance techniques, such as replication, n-modular redundancy at the object level, and others.

Although certain faults can be masked from the application, others require action on the part of the application and must be visible to it. Alpha manifests most faults to the user as exceptions. Alpha provides a block-structured exception handling facility. Examples of exceptions include thread break, failed object operation invocation, time constraint expiration, transaction abort, divide by zero, etc. Applications are provided with the facilities they require to clean up and exit gracefully in the face of asynchronous termination of an application routine--a general resource tracking facility makes it possible to maintain the consistency of application and system resources.

Alpha also provides mechanisms for dynamic reconfiguration, including location transparent naming and object migration. Alpha programming constructs support fault containment by providing separate, hardware-protected address spaces for every object and thread. The kernel is also protected, and runs at the supervisor protection level. The system attempts to minimize objects' exposure to faults by confining an object's representation to one node, as much as possible. Also, capabilities provide kernel protection of access to objects (i.e., objects cannot connect to other arbitrary objects to which they have not been granted access).

Threads, which are inherently distributed entities, recover easily from node failures. Alpha fully supports threads as a distributed programming abstraction, and provides thread repair (time-driven orphan detection and elimination, and restart at the earliest point of breakage) to recover from node failures.

Concurrent is teamed with General Electric's Advanced Technology Laboratories in a contract to investigate expert system based adaptive fault tolerance techniques for Alpha: the dynamic varying of the system's fault tolerance strategies in order to utilize the system's resources most efficiently in realizing the degree and kind of fault tolerance required by the system at any point in time.

4.2.1.1.12 Resource Management Interfaces

Alpha accepts responsibility for performing the management of global resources, both physical and logical, in the best interests (i.e., to meet the time constraints of) the entire distributed application suite. Forcing applications to perform their own resource management in a large, complex, distributed real-time system results in lower-performance, error-prone, recurring, inconsistent, and higher-cost solutions. Alpha's resource management is directed by application-specific policies. All resource contention in Alpha is resolved on the basis of the urgency and importance of the contending activities, in the context of the system's mission. Alpha seeks to manage resources so as to maximize (as nearly as possible) the value accrued to the system as a result of completing activities.

4.2.1.1.13 Synchronization and Scheduling Services

Scheduling is a specific case of resource (processor cycle) management, and Alpha performs it in a manner uniform with that of all other resources--i.e., using kernel level mechanisms which are employed by a system layer resource management policy. The initial policies provided include a series of best-effort algorithms which exploit time-value functions, plus conventional real-time (e.g., rate-monotonic) and non-real-time (e.g., SPT, FIFO).

Concurrent application activities are embodied as threads; each thread specifies a set of parameters that apply to a particular interval of its execution. These parameters determine a particular time-value function, and provide other information, such as an estimated computation time for the activity. Such time constraints may be nested, which allows finer-grained control and improved modularity. Time constraints may be varied dynamically at run-time. The resource management parameter interface is designed to support a wide variety of resource management policies. Because the information it provides is policy-independent, and based on actual requirements rather than artifacts of a particular resource management facility, it can be used across the system in any context.

Alpha's kernel therefore has observability of the actions of

activities, where they are currently executing, their time constraints and importance values, whether they are involved in atomic transactions (and which ones), which nodes they span, whether they are active or blocked, which resource and other activity dependencies need to be resolved.

Alpha's kernel provides semaphores and locks, from which more sophisticated synchronization constructs can be created at the system and application layers. The kernel also provides transaction mechanisms which may be used for global synchronization schemes.

4.2.1.1.14 System Initialization and Reinitialization Interfaces

Alpha's kernel provides primal objects and threads, and recovery operations for permanent objects. System level initialization and reinitialization services are being defined.

4.2.1.1.15 Time Services Interfaces

Alpha's kernel provides conventional time of day and both logical and physical time services (e.g., delay an interval or until a specific time), in addition to the time-value function facilities. A distributed global clock synchronization service is being planned at the system layer; provisions to support such synchronization are being devised for the communication software in the kernel.

4.2.1.1.16 Ada Language Support Interfaces

An Ada binding to Alpha's kernel has been defined but has not yet been implemented.

4.2.1.2 Additional Characteristics

4.2.1.2.1 Proprietary or Open

Alpha is portable, non-proprietary, and in tne public domain for U.S. Government use. Alpha licenses are available for commercial purposes.

4.2.1.2.2 Qualification as a Standard

Alpha is a DoD-sponsored, non-proprietary, multipurpose, portable OS for an application domain currently populated entirely by one- of-a-kind, special-purpose OS's: large, complex, distributed real-time systems. Because Alpha is still a pilot project and not yet commercially available, its exposure to potential users has been carefully controlled. Nonetheless, strong interest in Alpha is being expressed by the DoD contractor community for ground, surface and subsurface ship, air, and space applications. Interest in Alpha also has arisen from key factory automation applications such as automobile manufacturers.

4.2.1.2.3 Platform Flexibility

Alpha is intended to be highly portable. It is implemented primarily in C++ with a small amount of hardware-dependent code. The initial ports are to MIPS and 680x0 multiprocessors.

4.2.1.3 References

[CLARK88], [JENSE88a], [JENSE88b], [JENSE88c], [JENSE89], [NORTH87], [NORTH88a], [NORTH88b], [NORTH88c], [NORTH88d], [NORTH88e], [NORTH88f], [NORTH88g], [NORTH88h], [REYNO88a], [REYNO88b], [TRULL88].

4.2.2 Advanced Real-Time Operating System (ARTS) Survey Summary

The Advanced Real-Time Operating System (ARTS) is a distributed real-time operating system being developed under the Advanced Real-Time Technology Project (ART). The projects is a combined effort being conducted by Carnegie Mellon University (CMU), the Software Engineering Institute (SEI), and IBM. It is sponsored by the Office of Naval Research (ONR) under the Distributed Tactical Decision Making (DTDM) program. "The objective of the ART project is to develop theoretical foundations, distributed real-time operating system technology and programming language support that will facilitate practitioners in developing distributed real-time systems with an understandable, predictable, and maintainable behavior." [1, p. 2] ARTS synthesizes the results of real-time distributed systems research. Some of the areas the research group is investigating are real-time processor scheduling, real-time local area networks, real-time synchronization, real-time distributed database management and real-time cache management. As each of these research area matures they are incorporated into ARTS.

ARTS is concerned primarily with the needs of the target but does provide an interface to and support tools for the host machine. ARTS runs on the bare machine, not on top of an existing operating system. The tools run on an existing operating system and require X-windows, UDP and a C compiler. ARTS is presently under going development which makes it difficult to summarize it in terms of OSSWG interface requirements. Basic operating system functions and scheduling have been implemented on a single processor. Network scheduling was to be implemented by September 1989. Once network scheduling is implemented, the remote or distributed aspects of ARTS can be implemented, for example, the ability to do a remote invocation. Database features should be added within the next year. End-to-end scheduling of all system resources in a distributed environment and in real-time is still under development. For the most part, when reading this survey, interfaces that involve a single node are implemented and ones that involve remote nodes are part of the design but not implemented at present.

4.2.2.1 Operating System Service Classes:

4.2.2.1.1 General Requirements

ARTS was designed to support distributed, real-time applications which require a high degree of reliability. It provides bounded operating services times and context switching. At present, ARTS is implemented on a 68000 based architecture but is not dependent on this architecture. The design of ARTS is based on the object model promoting modularity, and extensibility. Being a research vehicle it lacks completeness. ARTS presently supports the C and C++ languages. Insufficient knowledge of ARTS interfaces prohibits comment on areas such as semantic consistency, cohesiveness and pragmatics.

4.2.2.1.2 Architecture Dependent Interfaces

ARTS has a UDP service which allows the target to interface with many of the commercial system that also support UDP. This is how the ARTS host and target communicate at present. There will be a special set of services provided for interface to a variety of networks, including Ethernet, token ring and IBM's Real-Time Communications Network (RTCN). ARTS does not interface to Navy standard computers such as the AN/UYK-44.

4.2.2.1.3 Capability and Security Interfaces

Characteristics of the object-oriented model used in ARTS support access control. The implementation of security in ARTS has not been emphasized heavily.

4.2.2.1.4 Data Interchange Interfaces

There are limited data interchange interfaces available.

4.2.2.1.5 Event and Error Interfaces

The extent of event and error interfaces is unknown.

4.2.2.1.6 File Interfaces

ARTS supports non-real-time access to UNIX files using the UDP mechanisms. An application can use all the basic UNIX files services such as naming and directory services and primitives, such as read, write, open, close, on files.

ARTS does not provide real-time file access at present. Access time to UNIX file using UDP over an ethernet is unpredictable and non-deterministic. The UNIX access to files is a convenience and not intended to work in conjunction with a real-time task. At present, all data needed by a task is stored in main memory. There is a real-time database effort but the

status is unknown.

4.2.2.1.7 Generalized I/O Interfaces

There is limit generalized I/O interfaces available.

4.2.2.1.8 Networks and Communications

ARTS places a strong emphasis on the ability to provide network and communication services. These services are needed to support the distributed features of ARTS. ARTS hopes to serve as a testing ground for a variety of networks such as token ring and SAFENET II and network protocols such as XTP and VMTP. The initial research on network services is completed and implementation has begun. There is an effort to provide coordination between processor and communication scheduling.

4.2.2.1.9 Process Management Interfaces

ARTS supports local and remote creation and destruction of objects. Some primitives include run, choose, block and kill. At the object and thread level there are additional primitives such as Object_Freeze and Thread_Fetch.

LPOS to LPOS communication is supported in combination with the object-oriented model. A variety of protocols are supported including XTP, VMTP and RTP. The object-oriented model hides details from the application. An application simply performs an invocation on an object. This feature has been implemented on a single processor and will be extended tc multi-processors in the future. The object model provides a consistent LPOS to LPOS communication service. Primitives include Request, AsyncRequest, AsnycRequestAll, GetReply, Accept, Reply, CheckRequest and CheckReply.

4.2.2.1.10 Project Support Environment Interfaces

The Project Support Environment for ARTS is typically a UNIX based machine. The application developer has all the usual support tools provided by UNIX plus some additional tools provided by ARTS. Down loading is supported through a shared partition on a file server. There are two tools provided with ARTS at present, the Scheduler 1-2-3 and the Advanced Real-Time Monitor/Debugger (ARM). "The goal of the toolset is to incorporate a system-wide scheduling analysis which includes communication and synchronization among real-time objects. Our schedulability analyzer, called Scheduler 1-2-3, is a X11-window based interactive schedulability analyzer for creating, manipulating, and analyzing real-time task sets. It employs methods ranging from closed form analysis to simulation to determine wnether a feasible schedule exists for a given task set and what the schedulable bound is for that set. ARM is also a X11-window based tool designed to analyze and visualize the runtime behavior of the target nodes in real time. ARM allows us to reach

into a remote target and view the scheduling events which are extracted using event traps in the Integrated Time-Driven Scheduler (ITDS) object of the ARTS Kernel." [2,p. 20]

4.2.2.1.11 Reliability, Adaptability, Maintainability Interfaces

Reliability is supported by the object model and ARTS ITDS. The Real-Time Invocation manager and Object/Thread Management mechanisms are responsible for fault detection, fault isolation and fault recovery. These mechanisms also guarantee that tasks will be accomplished in a timely fashion based on deadlines.

Adaptability is also supported by the object model. For example, since the ARTS scheduling policies are developed as objects, one policy can easily be replaced by another without necessitating changes in other parts of the system. This feature is extended to the application using the object model.

Maintainability is support by the object model which modularizes functions into manageable units and isolates data from corruption. Maintainability is also supported by providing time predictable services and a set of real-time tools. Programmers are better able to produce predictable systems that lack ad hoc characteristics.

4.2.2.1.12 Resource Management Interfaces

ARTS provides virtual memory management services that are modeled after Mach. Hard real-time tasks are allowed to be pinned in memory to avoid worst case context switching time. "By hard real-time task, we mean that the task must complete its activity by its "hard" deadline time, otherwise it will cause undesirable damage or fatal error to the system. The soft real-time task, on the other hand, does not have such a "hard" deadline, and it still makes sense for the system to complete the task even if it passed its "critical" time." [2, p. 6] ARTS does not support process migration at runtime. Instead it can move an object by stopping the object on one machine and restarting it with the appropriate parameter at another.

ARTS uses some of Sun's proprietary device driver software that has been modified to have real-time characteristics. ARTS supplies device drivers for ethernet, token ring, clock and tty.

4.2.2.1.13 Synchronization and Scheduling Interfaces

The Arts kernel supports a variety of real-time scheduling algorithms such as Rate Monotonic and Earliest Deadline First. It also supports traditional scheduling algorithms such as Round Robin and Fixed Priority. The ARTS Kernel allows the user to choose from a variety of policies and to switch policy at run time. The scheduling policy used by the scheduler to decide who gets a resource next is separate from the mechanisms to actually allocate and deallocate the resources. This allows new scheduling algorithms to be added to the system easily. The major effort has

been on developing a variety of CPU scheduling algorithms. More recently, disk, network and memory scheduling algorithms have been investigated. Selected algorithms will be added to ARTS in the near future. They have also been developing various scheduling techniques and tools that allow the programmer to determine the ability to schedule a given task set on a system wide bases. "By system-wide schedulability analysis, we mean that a system designer should be able to analyze or predict, at the system design stage, whether the given real-time tasks having various types of system and task interactions (e.g., memory allocation/deallocation, message communications, I/O interactions, etc.) can meet their timing requirements." [2, p. 1]

ARTS scheduling algorithms help to prevent service denial. ARTS guarantees that given a set of tasks or task that it will complete each job before it's deadline. Some of the algorithms that have not been implemented yet, such as best effort, guarantee that given a set of tasks that can not all meet their deadlines, that the best effort will be made to complete the most the system services.

4.2.2.1.14 System Initialization and Reinitialization Interfaces

There are interfaces for loading, initializing and shutting down ARTS.

4.2.2.1.15 Time Services Interfaces

ARTS supports a variety of timing services. Most important is the ITDS mentioned earlier.

4.2.2.1.16 Ada Language Support Interfaces

One of the major areas of the ART project is to develop a distributed real-time Ada runtime environment. The effort was to combine state-of-the-arts software engineering results with both the scheduling algorithms provided by the theory group and the services provided by the ARTS effort. The Ada work was being conducted by SEI. It is unknown whether or not this work has been integrated into ARTS.

4.2.2.2 Additional Characteristics

4.2.2.2.1 Proprietary or Open

ARTS is the result of research work that is funded by various DoD agencies. The source code is available upon request and the interfaces are open for implementation by all vendors.

4.2.2.2.2 Qualification as a Standard

ARTS does not qualify as a standard.

4.2.2.2.3 Platform Flexibility

ARTS is implemented on a 68000 based architecture and supports interfaces to several types of networks. The design does not preclude additional architectures.

4.2.2.3 References

[LEHOC86B] [TOKUD89]

4.2.3 ARTX Survey Summary

Ready System's Ada Real-Time eXecutive (ARTX) is designed to implement the critical "kernel" services of an Ada multitasking real-time Runtime System for embedded microprocessor applications (Motorola 68000 microprocessors family and Intel 80386 microprocessor). ARTX schedules the processor and allocates CPU time among a number of concurrent tasks; it also allocates blocks of available memory to these tasks and implements intertask communication and synchronization. Furthermore, it supports a full range of Ada semantic operations, including the complete Ada tasking model. Its real-time capabilities may be elaborated as follows:

* ARTX consists of deterministic algorithms with fixed, specified timing for task rescheduling, rendezvous calls and accepts, memory allocation, interrupt latency, and interrupts-off time.
* ARTX's timing is independent of system load. That means that, as system requirements change and more capabilities (tasks or other system objects) are added to the system, ARTX's timing is not affected.
* ARTX supports a fully preemptive scheduler so that the highest priority task in the system will always be executing.
* ARTX allows task priority to be changed at run-time.
* ARTX provides additional communication and synchronization primitives besides the standard Ada rendezvous. These primitives are accessed from the applications code via a packaged interface and include mailboxes, queues, semaphores, and event flags.
* ARTX supports two alternatives for servicing interrupts, using Ada rendezvous entries which are flexible or using Ada procedure calls which are fast but not as flexible.

ARTX is also upwardly compatible with Ready Systems' industry standard kernel VRTX32, so that application tasks written in other languages (C, Fortran and assembly language) can be easily integrated into the system without any changes.

4.2.3.1 Operating System Service Classes

4.2.3.1.1 General Requirements

ARTX supports mixed-mode applications where a number of tasks may co-exist in the same target system, may be written in a variety of languages (for example C, Fortran, Pascal and assembly language) and may interact in various ways through the underlying ARTX capabilities (this because ARTX is upwardly compatible with VRTX32).

4.2.3.1.2 Architecture Dependent Interfaces

No explicit features

4.2.3.1.3 Capability and Security Interfaces

No explicit features

4.2.3.1.4 Data Interchange Interfaces

Differences in type representation by various programming languages (Ada and C) within the ARTX environment are handled by the cross Ada and C compilers.

4.2.3.1.5 Event and Error Management Interfaces

ARTX uses Event Flag to signal occurrences of events to tasks. It also detects event overruns, and it provides synchronization features such as: a) a task can wait for one of several events to occur, b) a task can wait for a total of several events to occur, c) many tasks can be waiting for the same events to occur.

ARTX also utilizes the Ada exception mechanism for communicating error conditions. Exception handling features a complete traceback of the exception propagation. It provides information on the entire call chain of the exception, extending all the way to the originating Ada source line.

4.2.3.1.6 File Interfaces

4.2.3.1.6.1 Naming and Directory Interfaces

ARTX can be extended to provide file services by including the READY SYSTEMS' Input/Output File Executive (IFX). IFX is a file manager to provide file and directory handling functions for disk devices. The Disk I/O module of the IFX provides the MS-DOS compatible File Manager. The disk file operations are as follows: 1) formatting and initialization, 2) file management calls (open, close, create, delete, and rename files), 3) operations on directories (make and remove), 4) file locking, 5) volume mounting and dismounting, 6) I/O operations (transferring data to and from files).

4.2.3.1.6.2 Real-time Files

ARTX's IFX provides a buffer cache manager which reduces I/O operations to disks by maintaining a cache of frequently used sectors. Also, Interrupt Service Routines can make direct calls to IFX resulting in better serial operations.

4.2.3.1.7 Generalized I/O Interfaces

IFX's Stream I/O module handles I/O for byte-stream devices such as terminals, printers, pipes and other serial communications devices. The Stream I/O module consists of the Circular Buffer Manager and the Line Editor. The Buffer Manager is used for the high-speed binary communication between computers. The Line Editor is used with CRT terminals and printers.

4.2.3.1.8 Networks and Communications

ARTX can be configured to provide a multi-processor networked runtime environment by using its two companion components RTAda-MP and RTAda-Net. RTAda-MP for shared memory multiprocessing; and RTAda-Net for multiprocessor communication over local area networks.

RTAda-MP supports a multiprocessor system organized as a "cluster." A cluster consists of two or more processors with local memory global or shared memory; a system bus which connects the processors; ARTX and RTAda-MP on each processor; and Ada application software. It provides three layers of interprocess communication and synchronization services. The lowest layer is the Physical Layer, which provides unbuffered communication between nodes. The Channel Layer is built on top of the Physical Layer and provides node-to-node message passing mechanism (channel is a buffered, virtual connection between two Ada tasks on two different processors that allows the tasks to send and receive messages). The highest layer is the Remote Procedure Calls(RPC), which allow inter-processor procedure calls synchronously or asynchronously.

RTAda-Net allows ARTX based system to network with other systems that support TCP/IP and sockets (i.e., Sun/UNIX). For example, real-time applications in a RTAda-Net environment can uses ARTX to gather, control, and allocate real-time processes while a more familiar interface (such as VAX/VMS with TCP/IP, or Unix BSD) can be used for data storage, or as a system monitor in a non-real-time environment.

4.2.3.1.9 Process Management Interfaces

ARTX supports complete Ada tasking model. It provides system calls for creating and deleting tasks (processes), suspending and resuming tasks execution. It performs task scheduling using the preemptive priority-based techniques. Each task is assigned a priority when it is created. When more than one task is ready to run, ARTX always select the highest priority task. Optionally, ARTX schedules tasks of equal priority on a time-sliced basis.

When time-slicing is enabled, equal-priority tasks run to the user-specified time-sliced value in round-robin fashion.

4.2.3.1.10 Project Support Environment Interfaces

No explicit features

4.2.3.1.11 Reliability, Adaptability and Maintainability Interfaces

No explicit features

4.2.3.1.12 Resource Management Interfaces

Storage Management

ARTX's approach to dynamic memory allocation is based on the needs of real-time multitasking applications; that is speed and predictability. Because of all variable-block allocation schemes can, under certain conditions, produce unpredictable response times, ARTX allocates and release memory storage in fixed-size blocks and a free pool may be subdivided dynamically into partitions to obtain space efficiency. A task can minimize wasted memory by allocating from the partition with block size closest to the actual amount of memory it needs. Therefore, the real-time executive can manage the memory pool without searching overhead and without external fragmentation.

4.2.3.1.13 Synchronization and Scheduling Interfaces

ARTX provides Dijkstra counting semaphores for mutual exclusion to gain or relinquish exclusive control over a shared resource such as memory, and I/O device, etc. It also uses mailbox and queue for data transfer, synchronization and mutual exclusion. Event Flags is also used for intertask synchronization.

ARTX performs task scheduling using the preemptive priority-based techniques. Each task is assigned a priority when it is created. When more than one task is ready to run, ARTX always select the highest priority task. Optionally, ARTX schedules tasks of equal priority on a time-sliced basis. When time-slicing is enabled, equal- priority tasks run to the user-specified time-sliced value in round-robin fashion.

4.2.3.1.14 System Initialization and Reinitialization Interfaces

ARTX provides a system call for ARTX initialization.

4.2.3.1.15 Time Interfaces

Real-Time clock Services is based on the notion of a clock tick; a tick is derived from an interrupt generated by a hardware timer. ARTX maintains a clock counter which accumulates ticks; it increments the counter whenever an interrupt service routine issues

a system call. A high-resolution relative time clock like ARTX's is essential for embedded applications.

4.2.3.1.16 Ada Language Support Interfaces

ARTX supports complete Ada tasking operations (e.g. rendezvous, task creation and termination, task completion and abort, select, and delay). It also supports full range Ada semantic operations (e.g. exception handling, input/output and dynamic memory allocation).

4.2.3.2 Additional Characteristics

4.2.3.2.1 Proprietary or Open

ARTX is proprietary.

4.2.3.2.2 Qualification as a Standard

ARTX is upwardly compatible with Ready Systems' industry standard real time kernel for microprocessor VRTX32.

4.2.3.2.3 Platform Flexibility

ARTX is targeted to MC 68000 family microprocessors and Intel 386 microprocessor.

4.2.3.3 References

4.2.4 ATES 43 Survey Summary

ATES 43 is designed to operate in a complex of UYK 43 multiprocessor computers. It is an evolution from ATES, an operating system in use on the early CG 47 Class ships. ATES 43 will be used on the later CG 47 Class ships and on the DDG 51 Class ships. It provides a uniform fault tolerant message passing protocol among processes whether in the same or different computers. It will recover automatically from all single point hardware and software faults. It provides for pre-emptive priority based scheduling within a computer and limited priority based scheduling among processes in different computers. A more detailed description follows.

FUNCTIONS OF ATES 43

ATES 43 is one of the operating systems used in the AEGIS combat system. Specifically it is used in SPY-1D, C&D, WCS, UWS, ADS, and ACTS. ATES 43 has evolved to support the needs of the application programs it supports. It is tailored to these and to the UYK 43 computer on which it executes.

The following information on ATES 43 functions is taken from section 1 of the "AEGIS TACTICAL EXECUTIVE SYSTEM (ATES/43) USER'S

MANUAL, 30 June 1988, AEGIS SHIPBUILDING PROGRAM" with slight changes to improve readability in this document.

ATES 43 DESCRIPTION

Design and development of ATES 43 is based on several objectives that evolved from past user experience and AEGIS DDG mission requirements. These objectives include:

a. Provide an operating environment that responds in a predictable, user-controlled manner to stimuli from both program modules and the AN/UYK-43.

b. Manage and use fully the resources of the AN/UYK-43.

c. Protect itself from out-of-bounds parameters presented in requests for services, memory access, and hardware failures.

d. Rapidly recover from AN/UYK-43 hardware and computer program errors in order to met the high availability requirements for DDG AWS.

e. Provide for an external system clock to allow synchronization of programs and data in several computers.

f. Provide program protection during loading and initialization.

g. Provide a common interface for ATES 43 standard peripherals.

h. Provide for computer program architectures required by the tactical and simulation elements, including (1)multi-tasking; (2)repetitive and demand-driven tasks; (3)time-critical, non-time-critical, and time-sliced tasks; (4)Common Service Routines (CSRs); and (5)Subroutine. A computer program is composed of modules (processes), procedures, and data segments. ATES 43 manages the interaction among these components.

i. Provide a logical I/O capability to support the use of redundant data paths and the reconfiguration of a computer program to an alternate computer.

j. Provide for on-line program debugging, system/program performance measurement, and recording.

Whether or not objectives are met depends on how the user directs ATES 43.

BASIC OPERATIONS

ATES 43 provides for the basic Operator Communication Device (OCD) operations common to AEGIS Weapon System (AWS) computer programs, and it maintains the integrity of user programs by performing all the interrupt-state operations and executing all of the privileged instructions within the AN/UYK-43 computer. The user controls these basic operations with specification language statements at system build time, and with Executive Service Requests (ESRs), OCD inputs, and messages to ATES 43 task-state modules during system operation. Common operations provided with the OCD are:

a. Program loading b. Utilities c. Data recording d. Testing e. System resource monitoring

The user can provide additional OCD modes of operation.

A special program development operation, Split ATES 43, allows two ATES 43 programs to operate independently and simultaneously within one AN/UYK-43 enclosure. Each program uses

one CPU, one IOC, and up to five memory units.

4.2.4.1 Operating System Service Classes

The following paragraphs identify and briefly define the 16 functions provided by ATES 43. With the exception of IPL/43 (Initial Program Load), all the ATES 43 components are part of the operational program loaded by IPL/43.

4.2.4.1.1 General Requirements

The Utility Processing Function provides performance measurements and debugging support capabilities. Performance measurements include system and module statistics, executive trace information for debugging, and timing measurements. Debugging support includes inspecting and changing memory, setting instruction or data breakpoints, selecting data output on breakpoints, enabling/disabling the P-history file, printing load maps, and accepting a core dump. These tasks can be performed at an OCD or by messages from user modules.

4.2.4.1.2 Architecture Dependent Interfaces

No explicit features

4.2.4.1.3 Capability and Security Interfaces

No explicit features

4.2.4.1.4 Data Interchange Interfaces

No explicit features

4.2.4.1.5 Event and Error Management Interfaces

Error Processing and Recovery (EP&R) Function: The EP&R Function is responsible for processing all errors from computer equipment or program failures. It initiates reporting and recording of the errors and controls user-specified error recovery scenarios. These recovery scenarios are specified by the user at system build time.

The Interrupt Processing
Function receives and interprets all interrupts, whether they are generated as a result of software or hardware actions, and transfers control to the ATES 43 components for further processing.

The Diagnostic Approval Function approves or disapproves diagnostics recommended by the FTSRM Interface Module (FIM) or by the operator (via the System Resource Function) in accordance with the system operating conditions and the requirements provided by the user at system build time.

4.2.4.1.6 File Interfaces

The Data Recording Function extracts ATES 43 executive or user-selected, memory-resident data and records this data on magnetic tape for subsequent reduction and analysis by AEGIS Data Recording System (ADAR) or an ADAR-compatible program. The data recording function uses a common tape handler with device processing. The extraction points are specified by the user at system build time.

4.2.4.1.7 Generalized I/O Interfaces

The Device Processing Function provides the interface between (1)ATES 43 components and user modules, and (2) the ATES 43 standard peripheral devices. Currently, ATES 43 supports the following peripheral devices for the AEGIS Weapon System.
a. One magnetic tape unit (RD-358A) with up to 4 tape drives b. Two disk units (AEGIS UYH-3V, DDG modified) c. Two operator communication devices (OL-267) d. One printer (Data Products BP-1500/600)
The I/O Processing
Function is responsible for initiating all I/O operations except intercomputer operations. It processes all I/O interrupts received from the interrupt processor. The user specifies the channel and memory access attributes that are honored by the I/O processor.

4.2.4.1.8 Networks and Communications

Message Processing Function: The Message Processing Function is responsible for the distribution and routing of messages from one module to another within a computer and, via intercomputer processing, between computers. Message types, distribution and routing are specified by the user system build time.
The Intercomputer Communication Processing Function establishes intercomputer links, transmits and receives messages, gathers and scatters table data, reports link status, and processes interrupts received from the interrupt processor on intercomputer channels. The user logically enables communication on the channels and coordinates the processing, distribution, and routing code tables in each computer. These tables are specified by the user at system build time.

4.2.4.1.9 Process Management Interfaces

The Loading Function is performed by IPL/43, a stand-alone program. Once IPL/43 is boot loaded, it provides prompts to the OCD operator to load a Tactical Load File (TFL), an ATES 43 compatible load file, or a bootblock-formatted file. Subsequently, IPL/43 is used by the reload functions. Instruction and data segments are loaded into separate memory units as defined by the

user at System build time.

The Background Loading Function is responsible for loading a backup copy of the program. A backup copy of the computer program will be loaded while the primary copy of the program is running. To minimize conflict with the operation of the primary copy, this background loading from disk will be carried out one segment at a time with the disk services provided by the Device Processing Function.

The Command-Activated Remote Load (CARL) Function allows the user to load a computer program into one AN/UYK-43 computer from another (remote) AN/UYK-43 computer. The computer to be loaded must have an RS-449 interface between its Display Control Unit (DCU) and an I/O channel of the remote AN/UYK-43. ATES 43 provides the DCU and IPL/43 commands for loading a computer program from disk.

4.2.4.1.10 Project Support Environment Interfaces

Common Service Routines (CSR)s are a group of reentrant task-state routines that provide common services for ATES 43 and user modules within the computer program. Selected ATES 43 routines may be included in the computer program along with user-supplied routines at system build time. The ATES 43 CSRs include mathematical routines, format conversion routines, and ESRs. These CSRs are accessed by a subroutine call, which eliminates the ESR processing-time overhead.

4.2.4.1.11 Reliability, Adaptability and Maintainability Interfaces

No explicit features

4.2.4.1.12 Resource Management Interfaces

Memory Management Function: The Memory Management Function allocates and reclaims temporary storage shared by ATES 43 components and user modules. Memory management allocates blocks of memory as requested, and it reclaims the uncataloged blocks of memory when requested or when a module exists and the block is no longer needed. The size and distribution of temporary storage is specified by the user system build time.

The System Resource Monitoring Function provides the following services:

a. Monitors the status of the hardware modules that make up the computer enclosure, and informs the operator of any status change.

b. Notifies the AEGIS ORTS computer and a user-designated task-state module of requested hardware-module status changes.

c. Displays requested executive queue statistics on the printer during system operation.

d. Provides a means of software enabling/disabling hardware modules.

e. Carries out requests for diagnostics and tests on selected

system components.
f. Switches processing control to the backup copy of the program, or initiates a load from disk at the request of ATES 43 or the OCD operator.
g. Prints a physical-to-logical channel translation table for configured IOC channels upon request.
h. Enables/disables software program modules.
i. Prints module-fault and single-bit error data.

4.2.4.1.13 Synchronization and Scheduling Services

Scheduling Function: The Scheduling Function registers in the priority scheduling queue the requests for dispatching modules in accordance with designated priorities and throttling. It maintains this queue by incorporating scheduling request information in the queue and ordering the entries according to their assigned priority. One priority-scheduling queue serves both CPUs.

The Dispatching Function selects the highest priority request from the priority scheduling queue and transfers CPU control to that module at the specified entrance. (A module can have up to seven entrances. Since each can be scheduled individually and carries out different functions a module can be thought of as encompassing up to seven threads of control or processes. However two threads of control from a single module cannot be in execution concurrently.) This function also preempts the currently dispatched module when another module from the scheduling queue has a higher priority preemption level. The preempted module is suspended and will be dispatched again in its turn.

4.2.4.1.14 System Initialization and Reinitialization Services

The Resident
Initialization Function receives control from either IPL/43 or from the Error Processing and Recovery Function. Control is received from IPL/43 immediately after a program load or reload. Control is received from the error processor when a switch to the backup copy of ATES 43 is made. ATES 43 initializes itself and then dispatches user modules at their initialization entrances. Initialization is terminated as defined by the user.

4.2.4.1.15 Time Services Interfaces

The System Clock Processing Function controls the interface between the AN/UYK-43 and the AEGIS external clocks. It enables users to set and monitor master and slave clocks, and it contains a clock implant instruction in each IOC, which can be set to jump to a specified channel program every millisecond.

4.2.4.1.16 Ada Support Interfaces

No explicit features

4.2.4.2 Additional Characteristics

4.2.4.3 References

[AEGIS]

4.2.5 CAIS-A Survey Summary

The Common Ada Programming Support Environment (APSE) Interface Set (CAIS) revision A (CAIS-A) is a DOD standard for a kernel APSE (KAPSE) interface. It has been designated MIL-STD-1838A. It supersedes DOD-STD-1838 (commonly called CAIS-1 or sometimes just CAIS) and is a superset of it. As a KAPSE interface set, it defines interfaces for an APSE at the operating system level. It is Ada-oriented and seeks to provide a portability interface for project support tools. It is concerned with interface needs in the host, not the target, environment. It may be implemented either piggy-backed on an existing operating system or directly on a bare machine.

4.2.5.1 Operating System Service Classes

4.2.5.1.1 General Requirements

No other language bindings have been defined for CAIS-A, nor were any other languages explicitly taken into account during the definition of CAIS-A models or services.

4.2.5.1.2 Architecture Dependent Interfaces

CAIS-A includes interfaces for interfacing with other systems. These come in two forms.

The first is the ability to import and export data. This in turn comes in two forms. The first is interfaces which will import a file from an underlying OS to the CAIS-A database and, conversely, from a CAIS-A database to an underlying OS file. The second is interfaces which import and export information from/to a Common External Form (CEF). The CEF provides a canonical form in which to capture the information in a CAIS-A database and to recreate it at a new CAIS-A-based installation.

The second is the ability to interact with other systems through a gateway node. The other system may or may not be a CAIS-A-based system; it need only be capable of properly interpreting the protocols which can be utilized via the gateway node.

4.2.5.1.3 Capability and Security Interfaces

CAIS-A provides both discretionary access control (DAC) and mandatory access control (MAC). These have been designed in

accordance with the TCSEC and are intended to make it possible to achieve a B3 implementation. Because CAIS-A is trying to provide the minimal set of interfaces needed to assure portability of tools and because very few tools are concerned with both administration of security and being portable, CAIS-A provides no interfaces for authorization of access (e.g., logon) or for administration of security; it is assumed that the implementation will provide such services. In particular, there are no special CAIS-A interfaces for creating or manipulating groups (see below).

CAIS-A DAC is based on groups of users who can assume a given role. Roles are defined with respect to a given node and cover such rights as read, write, execute, read- attributes, write-relationships and control (i.e., the right to change the rights. When a process is executing on behalf of a user in a certain role, it can perform on that node those operations which are allowed by each of the granted rights. A group may be as small as one user and as large as all users on the system. A hierarchical structure of groups can be defined; any (sub)group can be included in more than one group. Each process has a default role, which is a default group under which it will always execute. In addition, a process can dynamically adopt and relinquish groups as it proceeds with its functions. Rights can also be denied as well as granted.

CAIS-A MAC interfaces are minimal, in keeping with requests from the security community that CAIS-A not dictate a particular security policy. Labeling of nodes and processes is provided for, but almost no semantics are defined. In particular, a SECURITY_VIOLATION exception is provided, but there are no specifics as to when a particular implementation must raise it. This is in keeping with the desire to provide a set of interfaces which could be implemented to achieve MAC, but which would not penalize those who wanted to implement it and use it under circumstances which did not require MAC precautions.

4.2.5.1.4 Data Interchange Interfaces

Although CAIS-A provides a canonical form for external data representation (see item 2 above), it is not at the level of representation of low-level data types such as discussed in the Reference Model for these services.

4.2.5.1.5 Event and Error Management Interfaces

CAIS-A utilizes the Ada exception mechanism for communicating error conditions. This is supplemented with a status code mechanism which provides more complete information, since many exceptions can be raised for multiple reasons.

CAIS-A also provides a mechanism by which the implementation can be requested to monitor an attribute on a given node. When the value of this attribute changes, a user-defined process is automatically invoked.

4.2.5.1.6 File Interfaces

CAIS-A does not provide a file system in the traditional operating system sense. Instead it has used an entity-relationship-attribute (ERA) model (called the node model) to unify the traditionally separate worlds of data (file) services, process management and input/output. In the node model, every important aspect of the system (e.g., files, directories, processes, devices, groups) is represented as a node with attributes, relationships and possibly contents. Paths can be traversed through the system by following the relationships. Names are applied to these relationships, so any node in the system can be identified by at least one pathname (i.e., the name that results from the concatenation of the relationship names as the relationships are traversed).

Using this node model, the user can store data according to a number of paradigms. One common usage is to mimic a hierarchical file/directory structure as is often found in conventional file systems. This hierarchical "backbone", however, is supplemented with the added benefits of a full ERA system.

Because of the nature of relationships in general and some particular features of the CAIS-A node model, every node may have, in addition to its primary pathname, any number of other pathnames. Thus, one process may access a node by one name and another process may access the same node by a completely different name. File (node) sharing is simple.

CAIS-A has no provisions for real-time files. All normal primitives for file (node) modification are available, including creating, opening, closing, reading, writing, renaming, copying and deleting. These services are largely provided by I/O packages which completely mimic Ada's Chapter 14 I/O.

4.2.5.1.7 Generalized I/O Interfaces

All forms of CAIS-A I/O are very similar. They are unified by the I/O Model which governs how file/device nodes can be connected with processes to achieve I/O. Packages are provided which are oriented towards certain classes of devices (e.g., terminals, magnetic tape) and services provided follow in the style of the Ada Chapter 14 I/O.

4.2.5.1.8 Networks and Communications

CAIS-A can be implemented as a distributed system. This can be done transparently to the applications system; in addition, CAIS-A provides interfaces which an application can use to direct some aspects of distribution and networking.

There are no services for administering or controlling the network itself. There are services for directing (actually, suggesting) that a process be run on a particular processor or that a file node be created on a particular device. Some status about the configuration can also be ascertained. As the configuration

is represented by nodes in the node model, the naming provided is exactly like that of the rest of the node model.

4.2.5.1.9 Process Management Interfaces

CAIS-A provides a full range of normal process management primitives: create, invoke, abort, suspend, resume and various status inquiries. Inter-process communication is provided in a manner that is uniform with the overall I/O model, making communication between processes indistinguishable from communication between a process and a file node. Processes can be invoked (in which the parent awaits the completion of its child) or spawned (in which the parent proceeds in parallel with its child). Processes occur in trees, and some primitives can be applied to the entire process tree.

4.2.5.1.10 Project Support Environment Interfaces

CAIS-A does not include any explicit project support environment services. It is possible to use CAIS-A features to add packages which provide such features (e.g., host- target communications). CAIS-A does not include any special interfaces which would support the needs of a debugger (e.g., break or single-step).

4.2.5.1.11 Reliability, Adaptability and Maintainability Interfaces

No explicit features

4.2.5.1.12 Resource Management Interfaces

CAIS-A provides no memory management services.

CAIS-A's resource model does address the needs of device management services. Through its representation in the node model, the resource model can be used to provide information and services related to device allocation, availability, and control.

4.2.5.1.13 Synchronization and Scheduling Interfaces

Synchronization is provided by CAIS-A through the use of two mechanisms. One is local node-level synchronization through the use of node handles, which use the intents which a process has declared with respect to a particular node to coordinate its access with that of other processes. This level of synchronization is used every time a node is opened. The second is transactions, which provide the ability to treat a sequence of operations as an atomic action with the guarantee that either all the operations will succeed or the system will be left in state in which none of the operations occurred. This can be used as a by- product of process invocation or as an independent action.

CAIS-A considers scheduling to be the responsibility of the

implementation and does not reflect any particular scheduling approaches at the level of the interfaces. It does not involve itself with task scheduling, leaving that to the particular compilation/run-time system being used.

4.2.5.1.14 System Initialization and Reinitialization Interfaces

These are considered to be the responsibility of the implementation and are not covered in the CAIS-A interfaces.

4.2.5.1.15 Time Interfaces

CAIS-A relies on Ada semantics for some time services. No others are provided.

4.2.5.1.16 Ada Language Support Interfaces

CAIS-A was defined in accordance with the concepts of Ada to the greatest extent possible. It does not address Ada tasking; it assumes that the Ada compilation system takes care of Ada semantics and deals only with processes, which are defined to be executing Ada programs. The ARTE is independent of CAIS-A. Likewise, CAIS-A (the standard) does not get involved with exception propagation, interrupt-to- task mapping, priority of tasks or rendezvous. The CAIS-A process model does not mimic the Ada tasking model.

4.2.5.2 Additional Characteristics:

4.2.5.2.1 Proprietary or Open

CAIS-A is fully open for implementation by all vendors. It is the product of a joint DoD/industry team.

4.2.5.2.2 Qualification as a Standard

CAIS-A is a DoD standard sponsored by the Ada Joint Program Office. It has been formally balloted among the services and other federal agencies, in accordance with the rules for establishing a DoD standard. This balloting included several hundred reviewers from industry as well as some recognized interested organizations such as EIA and SIGAda.

4.2.5.2.3 Platform Flexibility

CAIS-A is intended to be implementable on any of a wide variety of architectures which include machines from virtually any vendor.

4.2.5.3 References

MIL-STD-1838A. The Common Ada Programming Support Environment (APSE) Interface Set. 6 April 1989

4.2.6 Clouds Survey Summary

The Clouds project was initiated at the Georgia Institute of Technology in 1979. Since then, it has received major funding from NSF, NASA, and RADC. Currently, the Clouds project exists as part of a larger NSF-sponsored project called DARE (for Distributed Application Research Environment).

Clouds is a distributed operating system for a cluster of general purpose computers interconnected by a medium to high speed local area network. Its goals may be elaborated as follows:

- o Reliability and fault tolerance: In the beginning, the primary design goal of the Clouds distributed operating system was the support of reliable, fault-tolerant distributed computing. While reliability/fault tolerance remains as a major goal, the emphasis has shifted, as explained in the next paragraph.
- o Object/thread model: The object/thread programming model was originally conceived as a means to an end, the end being reliable and fault tolerant distributed computing. However, it has become an end in itself; the support and exploitation of this advanced programming paradigm is now the overriding theme of the Clouds research. Research topics include operating system support for objects, replication and consistency management using objects in a distributed environment, and programming language/methodology/tools support for programming distributed applications using objects.
- o "Minimalist" philosophy for distributed operating system design and implementation: The Clouds operating system supports a minimal set of functions necessary to run a distributed system. The object/thread model provides a structured persistent memory, doing away with the need for long-term storage in the form of a file service and complicated I/O system. The operating system does not incorporate services such as printing, databases, and graphics, either, since these can be effectively implemented as user level applications. The kernel of the Clouds operating system is also minimal, in keeping with this philosophy.

4.2.6.1 Operating System Service Classes:

4.2.6.1.1 General Requirements

The first language being supported on the Clouds system is C++. The language has been somewhat modified (by adding keywords) to support entry points, segmented data, persistent data, and permanent and temporary memory allocation. The C++ programs define objects (and not processing). Support for single inheritance is complete, and support for multiple inheritance is being designed. The language defines the inheritance scheme, as Clouds does not define inheritance (but does have the mechanisms to implement inheritance and sharing effectively).

4.2.6.1.2 Architecture Dependent Interfaces

The Clouds researchers plan to support Clouds-UNIX interoperability, of two distinct flavors. First, Clouds services should be made available to UNIX users and programs, through a Clouds library on UNIX, in a way that would enable a cluster of Clouds machines to serve as a back-end distributed system to UNIX workstations. Second, established UNIX services (e.g., mail, text processing, etc.) should be made available to Clouds users, through a "UNIX gateway."

4.2.6.1.3 Capability and Security Interfaces

Clouds utilizes capabilities for object naming. Each Clouds object is named and accessed by its capability, which is globally unique and location-independent.

At this point in time, protection is not a goal of the Clouds project. Therefore, although capabilities could be utilized for protection as well as for naming, they currently are not being utilized for this purpose.

4.2.6.1.4 Data Interchange Interfaces

Not incorporated into the operating system proper as an explicit service. However, it may be viewed as being supported by adherence to conventions.

4.2.6.1.5 Event and Error Management Interfaces

Object-based error handling and event management will be provided by Clouds. However, the exact nature of these services is not completely defined at this time.

4.2.6.1.6 File Interfaces

A conventional file can be viewed as a special case of a Clouds object, namely one with file data in its data space and file operations, such as read and write, which can be invoked by threads.

However, in the Clouds programming paradigm, the need for having files goes away. Programs do not need to store data in file-like entities, because they can keep the data in the (permanent) data space of objects.

Just as Clouds does not have files, it does not provide user-level support for file (or disk) I/O. The system creates the illusion of a huge virtual memory space that is permanent, and thus the need for using disk storage form a programmer's viewpoint is eliminated.

4.2.6.1.7 Generalized I/O Interfaces

No explicit features

4.2.6.1.8 Networks and Communications

Clouds provides two modes of interprocess communication, both stemming from the Clouds paradigm of global persistent objects, which can be shared on a system-wide basis. In particular, objects can be invoked using either one of two mechanisms: remote procedure call (RPC) or distributed shared memory (DSM). Using RPC, the thread migrates to the home site of the object and executes there; using DSM, the invoked object is demand paged to the site of the invoking thread. The mechanisms have orthogonal advantages and can be chosen for optimum performance.

4.2.6.1.9 Process Management Interfaces

Clouds adopts the object/thread model. The object serves as an abstraction of storage, and the thread as an abstraction of computation. Object invocations serve as the integrating mechanism. These abstractions are summarized below:

- Object: In Clouds, an object is an instance of an abstract data type. It is a passive entity, in particular, a persistent virtual address space. It is used to encapsulate all data, programs, devices, and resources.
- Object Invocation: Objects are accessed via (and only via) object invocations, to operations defined on the objects.
- Thread: The thread is the active entity in Clouds, the unit of computation and concurrency that is used to execute the code in objects. Threads traverse objects, independently of machine boundaries, via object invocations. Threads are implemented as lightweight processes. A thread that spans machine boundaries is implemented by several processes, one per machine.

Clouds is a distributed operating system. Its kernel is replicated at each node that participates as part of a Clouds distributed system. The kernels implement Clouds IPC. System services, provided by system objects, are invoked using Clouds IPC.

4.2.6.1.10 Project Support Environment Interfaces

No explicit features

4.2.6.1.11 Reliability, Adaptability, and Maintainability Interfaces

Clouds is being designed to offer a range of consistency-preserving mechanisms, from "best effort" to absolute consistency.

The consistency preserving mechanisms are based on attaching consistency labels to the operations declared in the objects. The

labels allow the operations to update the objects with (1) transaction-like semantics, for preserving inter-object consistency of data, (2) locally atomic semantics for preserving the consistency of data locally within one object, or (3) best-effort semantics like the way processes in conventional systems update memory and files.

Ongoing research is addressing fault tolerance using replicated data and computation.

4.2.6.1.12 Resource Management Interfaces

Storage Management Interfaces

In Clouds, emphasis is placed on the object as an abstraction of storage. The object is viewed as unifying the concepts of file space (long-lived storage) and memory space (volatile storage, but essential for computation), by providing a persistent virtual address space. Since objects provide permanent storage, the need for a traditional file system is eliminated; the file system is replaced by object memory. Object memory is stored on disk and demand paged. The demand paging happens with storage on the local machine, if the invocation uses RPC. The demand paging occurs over the network if the invocation uses DSM.

General Resource Management Interfaces

The Ra kernel manages the low-level scheduling of threads, demand paging, and segment and memory allocation. All other resource management tasks are done at the higher level through system objects. Currently, the system objects under implementation will do object management, task management, naming, and partition management. More will be implemented as the system evolves. One of the points of the Ra approach is to be flexible and avoid being locked into any particular resource management scheme.

4.2.6.1.13 Synchronization and Scheduling Interfaces

No explicit features

4.2.6.1.14 System Initialization and Reinitialization Interfaces

No explicit features

4.2.6.1.15 Time Interfaces

No explicit features

4.2.6.1.16 Ada Language Support Interfaces

Ada, as well as some other languages, will be supported on Clouds at a later date.

4.2.6.2 Additional Characteristics

4.2.6.2.1 Proprietary or Open

Open, since it is a government-sponsored academic research project.

4.2.6.2.2 Qualification as a Standard

The Clouds project is directed at exploring a non-conventional methodology or paradigm for building operating systems. It is meant to demonstrate the utility and effectiveness of the new methodology. It is based on a minimalist philosophy, to allow for customization. Its focus, at this point in time, is still on the fundamental paradigm that it advocates. As the system matures, the researchers will likely explore higher level system services in more detail.

4.2.6.2.3 Platform Flexibility

Clouds defines a methodology that advocates unifying a distributed system using a set of global, persistent address spaces along with necessary structuring, naming, and consistency support. It couples long term storage with addressable memory, and decouples processing from storage. Clouds also demonstrates the following:

- This environment can be built using a structured, portable minimal kernel and plug-in system services.
- Most operating system services can be handled at the application level, allowing for customization.
- Management of shared persistent memory can be handled effectively in a distributed system.
- Consistency of persistent memory can be handled.
- Fault tolerance can be achieved and fine tuned through replicated objects and replicated computations.

The current prototype runs on a set of Sun 3/60 machines on an Ethernet. The design does not preclude any form of machine from multiprocessors to embedded systems or any form of networking.

4.2.6.3 References

[BERNA], [DASGU88], [GIT86], [PITTS88]

4.2.7

Cronus has been under development at BBN Laboratories since 1981. It is sponsored by the Rome Air Development Center (RADC).

Cronus is an environment to support coherent integration of heterogeneous computer systems. Typically, the computer systems fall under a common administrative domain, and are interconnected by one or more high-speed local area networks. The computer systems may also be interconnected by wide area networks, via an internet (such as the DARPA Internet). Each set of computer systems is called a "cluster." The initial focus of Cronus has been on intracluster communication and cooperation; however, more recently, consideration has been given to intercluster aspects. The goals of

Cronus may be elaborated as follows:

- The ultimate goal of Cronus is to integrate heterogeneous computer systems into an effective general-purpose distributed computing environment for the development and execution of large-scale applications.
- Heterogeneity is the key concept. The hallmark of Cronus is its support of heterogeneity -- of both hardware and software resources. The motivation for this emphasis is threefold: (1) to allow applications and users to take advantage of the unique functionality offered by various hardware and software resources, (2) to allow existing software to continue to be used, and (3) to allow familiar computing environments to continue to be used.
- In particular, Cronus is designed to interoperate with, rather than to replace or totally encapsulate, constituent (i.e, native) operating systems.
- In addition to heterogeneity, the Cronus project places major emphasis on providing comprehensive support for large-scale distributed application development.

The Cronus approach is to introduce layers of software on top of constituent operating systems (or, in some cases, on bare hardware). Cronus is based on the object model; each system resource is a typed object, and is accessed through operations defined by the type. The object model provides an extensible architecture, in that application developers can cast application-specific resources in terms of new object types, which can be defined as subtypes of existing types.

Cronus supports heterogeneity by serving as a by- passable layer of abstraction between application programs and constituent operating systems. Through this approach, application programs gain access to a coherent, uniform (object-oriented) system interface, regardless of computer system base; however, they also retain conventional access to constituent operating system resources and services.

The Cronus distributed operating system consists of the following components:

- Cronus kernel: The Cronus kernel supports the Cronus object model. Namely, it implements the basic abstractions of object, operation invocation, and (Cronus) process, as defined below. It must be installed and run on each host participating in the Cronus distributed system. Typically, it is implemented as an application process of the constituent operating system.
- Cronus system services: Cronus system services provide the traditional operating system services, plus additional services specifically designed for the support of distributed application development. Each system service is implemented by one or more manager processes (i.e., servers), which run above the Cronus kernel as Cronus processes. Current system services include an authentication service, a catalog service, a configuration service, a file service, and a type definition service.

The distributed computing architecture supported by Cronus includes the following components as well:

- Application services: An application service is one or more

processes developed by application programmers to manage the resources that make up applications. An application is typically composed of several services responsible for several different object types.

o Clients: Clients are processes that use services. While any service may act as a client to another service, most clients are processes that interact directly with users, such as user commands, utilities, and application-specific graphical user interfaces.

4.2.7.1 Operating System Service Classes

4.2.7.1.1 General Requirements

The fundamental assumption underlying Cronus programming support is that large-scale applications will be developed in accordance with the object model, just as Cronus itself is. Under this assumption, the key to application development is the definition of new object types to represent application-specific resources and the development of new object managers to embody the newly defined object types. Therefore, Cronus programming support focuses on automating the process of developing new object managers. In particular, Cronus seeks to relieve the application developer's coding burden through the use of a non-procedural program development specification language. Cronus takes non-procedural specifications of a new object type, and automatically generates code for skeletal object managers (including multitasking for concurrent operation processing, message parsing and validation, access control checks, operation dispatching, data conversion between canonical and system-specific data representations, and stable storage management), as well as for RPC client stubs. The code generation process relies upon the Cronus libraries; the skeletal object managers incorporate procedure calls to Cronus library routines for many functions (e.g., data conversion between the canonical and system-specific representations of common data types). The application developer completes the object manager by providing routines that implement the operations defined by the new object type.

Cronus programming support also includes (1) extensive subroutine libraries, including interprocess communication routines, data conversion routines, and RPC interfaces to Cronus objects; (2) a set of user commands; (3) a set of operator commands; (4) operations inherited by all objects, for access control, monitoring and control, debugging, and replication and migration support; (5) a program to be used in conjunction with a local debugger, to assist in object manager debugging; (6) source management control software; and (7) a bug tracking facility.

Programming support was initially focused on C, but it is now being extended to Common Lisp and Ada. Application components have also been written in FORTRAN.

4.2.7.1.2 Architecture Dependent Interfaces

No explicit features

4.2.7.1.3 Capability and Security Interfaces

In Cronus, protection is achieved through access control lists. The access control list for an object specifies which users or groups of users have which access rights to the object. Privileges associated with access control lists can be defined separately for each object type. These privileges are specified by the application developer, allowing access controls to be customized for each type. Authentication (of the identity of a user) is implemented by an authentication manager, which subjects a user to a password-based authentication procedure upon login.

Multilevel security was investigated in a research project, the Secure Distributed Operating System (SDOS) Project. Among the conclusions of the project was the following [Casey 87, p.19]: "Thus, the host operating system(s) on top of which SDOS [i.e., secure Cronus] is implemented must have a minimum of a B2 rating, and ratings of B3 or A1 are more desirable." GEMSOS, a product of Gemini Computers, Inc., of Carmel, California, was selected as the best candidate for serving as a multilevel secure constituent operating system.

4.2.7.1.4 Data Interchange Interfaces

Cronus uses the technique of canonical data representation to solve the problem of data interchange in a heterogeneous computing environment. Programs process data in formats directly supported by the systems on which they are implemented. When data is transferred to another network component, it is encoded into a canonical form using appropriate conversion routines. The reverse process takes place on the receiving end. Cronus takes non-procedural specifications of a new object type, and automatically generates code for data conversion between canonical and system-specific data representations.

4.2.7.1.5 Event and Error Management Interfaces

Cronus routines that detect errors generally signal a failure by returning a special value distinguished from the set of normal return values. Routines returning numeric results can return ERROR, and routines that return pointer can return NULL. Before returning, these routines generally set the "ErrorBlock" -- a global structure in the program that records error conditions. Then, the calling routine has the option of attempting error recovery action using the information it find in the ErrorBlock, resetting the ErrorBlock with its own interpretation of the error, or simply returning the error indication provided by the lower-level routine.

4.2.7.1.6 Files Interfaces

Cronus provides a file system for storing information just

as do other operating systems. Cronus files are objects, so they are accessible through the same object-oriented, location independent IPC facility as are other Cronus system entities. Files in the Cronus files system are accessible from any and all hosts in the Cronus cluster automatically. Cronus will locate a file and direct operations to that file's object manager transparently, hiding the distributed nature of the file system, and providing an interface to the application program or user that is simple and powerful.

Though, different file types are implemented as different object types, the object-oriented nature of Cronus allows all of these file types to respond to a common set of operations through the mechanism of inheritance. Currently, two Cronus file types are available:

- o COS Files (Constituent Operating Systems Files): ordinary local host operating system files that have been made into objects and are accessible from anywhere within Cronus.
- o Reliable Files: characterized by special facilities for synchronizing access by multiple simultaneous readers and writers, by enhanced read and write operations that can simplify application programs, and by enhanced survivability in the face of system failures.

4.2.7.1.7 Generalized I/O Interfaces

Devices, such as line-printer, tape-driver, or terminal, are integrated into the Cronus system as sub-types of a generalized I/O object, which supports a generalized set of I/O operations. File-like interfaces for device I/O are supported for most devices.

4.2.7.1.8 Networks and Communications

Network Control and Status

The monitoring and control system (MCS) for Cronus includes monitoring and control of hosts and of the Cronus functions on these hosts, of the network substrate and of gateways.

Interprocess Communication

Cronus interprocess communication (IPC) is designed to support operation invocations from clients to object managers, where the invocations can be synchronous or asynchronous, and can have one or many targets. It is implemented as a series of layers.

At the lowest layers, collectively referred to as the network layer, are standard data communication protocols, which are typically implemented by the constituent operating systems. Currently, Transmission Control Protocol (TCP), User Datagram Protocol (UDP), Internet Protocol (IP), and Ethernet are utilized. However, other protocols could be substituted easily.

Above the network layer is the layer designated as the IPC layer. This layer implements three communication primitives: Invoke, Send, and Receive. In a typical scenario, Invoke would be used by a client process to invoke an operation on an object.

Using Invoke, the client references the object by name (not location, thereby ensuring host-independent, network-transparent access to objects), and this causes a message to be sent to the process serving as object manager of the target object. The object manager would retrieve the message from its message queue via the Receive primitive, perform the requested operation, and then send a reply to the client via the Send primitive. The operation would actually be performed by a lightweight process (or task, in Cronus terminology) created by the object manager; thus, operations can be performed concurrently. Finally, the client would receive the reply via the Receive primitive. The separation of the client's Invoke from the subsequent Receive allows for asynchrony and concurrency. It should be noted that the Send is simply an optimization of the Invoke. It allows a message to be sent directly to a process, instead of to the process manager.

Above the IPC layer is a layer designated as the message encodement layer. This layer is responsible for encoding and decoding messages, using canonical (system- independent) data representations. Cronus defines canonical data representations for many common data types and structures. It also offers extensibility by supporting the creation of new canonical types from existing ones.

At the highest layer is a protocol designated as the Operation Protocol. This layer defines a set of standards for interpreting messages between clients and managers, and supports a synchronous remote-procedure-call-like (RPC-like) programming interface for operation invocation.

Naming

Cronus has a two-level naming system. At the high level is a hierarchical symbolic name space. At the low level is the flat name space of Unique Identifiers (UIDs). A UID is a 96-bit object identifier, which is guaranteed to be unique over all objects over all time within a cluster; sixteen bits of the UID specify the object's type, and the remaining bits establish uniqueness. The Cronus catalog, which is implemented as a distributed entity by the catalog managers, provides the mapping between symbolic names and UIDs.

4.2.7.1.9 Process Management Interfaces

Since Cronus is based on the object model, the basic abstractions are objects and operation invocations. To implement the object model, the Cronus kernel introduces the process as a kernel-supported object type. Thus, the basic abstractions of Cronus are the following:

- Object: In Cronus, an object is an instance of an abstract data type, where a type can be defined as a subtype of a parent type, and hierarchical inheritance is supported. Objects are passive entities.
- Operation Invocation: Objects are accessed via (and only via) operation invocations. (This abstraction is inherent in

the object abstraction.)

o Process: Processes are the active entities in Cronus. They are used to implement object managers, as well as application programs that execute on Cronus. An object manager is the entity that is responsible for manipulating all of the objects of one or more given types on a given host using the operations defined by the types. The Cronus system managers are simply Cronus-provided object managers, for Cronus-defined object types. The Cronus process abstraction corresponds to the process abstraction found in conventional operating systems, and is typically implemented as a constituent operating system process that executes in user space.

Cronus Kernels communicate with one another using the Cronus Peer to Peer protocol. Reliable delivery of datagrams is guaranteed through the use of TCP as the transport mechanism. The Peer to Peer Protocol includes a specification for establishing TCP links between Kernels and for closing them down. A provision is also included to send low effort datagrams and broadcast/multicast messages between Kernels using UDP datagrams.

4.2.7.1.10 Project Support Environment Interfaces

The Tropic (Transportable Operation Interface for Cronus) program allows a user to invoke arbitrary operations on Cronus objects (e.g., interactive debugging of an object manager on the target system).

4.2.7.1.11 Reliability, Adaptability and Maintainability Interfaces

Cronus supports object migration and object replication. With respect to replication, the Cronus project has recently adopted the philosophy of application- specific replication management. Namely, Cronus has progressed from an inflexible weakly consistent replication strategy to a flexible "version voting" replication strategy. In the weakly consistent replication strategy, updates were propagated on a best-efforts basis, and object managers would periodically (e.g., upon their host coming back up after being down) utilize Cronus-provided mechanisms to bring their copies up to date. In the version voting replication strategy, version vectors (one for each replicated object, giving host location and version number pairs) are used to keep track of copies and consistency, and read and write quorums can be set to provide the application-desired balance between availability and consistency.

Cronus delivers replication support to application developers through its object manager programming support tools. When specifying a new object type, the application developer defines a replication policy by selecting a Cronus-supported replication strategy and then specifying values for the parameters of the selected strategy.

Two replication strategies are now available. The first,

referred to as version voting, mandates application- specified vote quorums to perform read and update operations. Version vectors are used to detect and correct inconsistencies. The second replication strategy, referred to as weak consistency, it that provided by precious versions of the Cronus manager development tools. BBN will be looking into other algorithms for replication. Based on the object type definition, Cronus automatically generates the code that implements the specified replication policy.

Cronus can also dynamically locate objects on invocation, ensuring that clients will always be able to access a copy of an object (providing one is available).

Atomic transaction support is being investigated in the context of distributed database management systems, as a part of the Cronus Distributed Database Management System Project.

4.2.7.1.12 Resource Management Interfaces

In Cronus, global resource management is approached according to the principle of policy/mechanism separation. That is, Cronus provides mechanisms, and the mechanisms enable object managers to cooperatively enforce object type-specific policies. The mechanisms include: (1) the ability of object managers to query the status of their peer object managers, one of which must be installed at each host where objects of the given type exist, (2) the ability of object managers to redirect requests to peer object managers, and (3) the ability of applications to indicate preferred hosts. These mechanisms support high-level resource management; low-level resource management is performed by the constituent operating systems. These mechanisms have been used in several services to implement specific management policies, such as dynamic load balancing during Cronus file creation.

4.2.7.1.13 Synchronization and Scheduling Interfaces

Cronus provides concurrency control for sensitive regions via the Start Concur and EndConcur routines. These are used in an object manager to bracket critical sections of code and provide concurrency control for accesses to the object database. This permits multiple operations to access the same object without fear of creating any inconsistencies in that object or it object database.

Semaphores are provided to control access to critical data structures. Since Cronus object managers may consist of multiple threads within a single process, semaphores can be used when data whose integrity must be insured may be shared by one or more tasks. The semaphore package causes threads to sleep when they are waiting to enter critical sections, and be awakened when the resource they are awaiting is available.

Cronus object managers use a coroutine package which provides a C programming interface for priority-ordered, non-preemptive multiple threads of execution within a single constituent operating system process, along with mechanisms for synchronization and mutual exclusion of critical sections.

4.2.7.1.14 System Initialization and Reinitialization Interfaces

Cronus object managers are asynchronous independent processes which are started by the system when it boots or by other processes (users).

4.2.7.1.15 Time Interfaces

Via constituent (i.e., native) operating systems.

4.2.7.1.16 Ada Language Support Interfaces

Programming support is now being extended to Ada, as noted under General Requirements.

4.2.7.2 Additional Characteristics

4.2.7.2.1 Proprietary or Open

Sponsored by Rome Air Development Center (RADC).

4.2.7.2.2 Qualification as a Standard

Essentially, the Cronus approach to dealing with heterogeneity is to introduce a layer of "standardization," in the form of the Cronus environment, between constituent operating systems and application programs. The issue is the utility, effectiveness, appeal, and acceptability of the Cronus environment. To date, Cronus has received only isolated support outside of BBN.

4.2.7.2.3 Platform Flexibility

Cronus has achieved high portability. It is written in the C programming language. Machine-dependent code is confined to a few modules. Cronus has been ported to a new machine in as little as two man-weeks.

Cronus implementations exist for the following systems: DEC VAX with VMS, Ultrix, and BSD Unix; SUN 2,3,4 and Sun 386i with Sun UNIX; MASSCOMP with RT UNIX; Symbolics Lisp Machine with Genera; and IBM PC/AT with SCO Xenix. Cronus implementations are planned for multiprocessor architectures.

4.2.7.3 References

[BBN88a], [BBN88b], [BBN88c], [BBN88d], [BERET85a], [BERET87], [CASEY87], [DEAN87], [DEAN88], [GURWI86], [SCHAN85], [SCHAN86a], [SCHAN86b], [VINTE87], [VINTE88]

4.2.8 43R88 Survey Summary

The AN/UYK-43 Runtime Support Software (43RSS) is a United States Navy standard operating system for the U.S. Navy standard AN/UYK-43 computer. 43RSS is an outgrowth of SDEX/7 and the ModX2 Common Program, a U.S. Navy standard operating system for the AN/UYK-7 computer. 43RSS is a real time tactical executive operating system that accommodates all the physical and functional features of the AN/UYK-43. Thirteen components currently make up 43RSS:

a. Standard AN/UYK-43 Executive (SDEX/43) - basic control for real time system operation by coordinating the use of the AN/UYK-43 resources through services such as task scheduling, dispatching and initial interrupt handling.

b. Common Systems - memory storage for routines and data which are available for use by all system software.

c. Common Peripheral - a general purpose, common interface between system software or the operator and the standard shared peripheral devices.

d. File Handler - an efficient means for users to create, manage, and maintain files on disk and magnetic tape.

e. Dynamic Modular Replacement (DMR) - AN/UYK-43 memory management and loading and deleting of system software.

f. Fault Tolerant and System Reconfiguration Module (FTSRM) Interface Module (FIM) - system access to and coordination of the error processing and isolation features embedded in the AN/UYK-43 FTSRM and on-line diagnostics firmware.

g. Fault Acceptance Module (FAM) - management of system software configuration loads, reconfiguration, and diagnostics test decisions.

h. Host Interface Adaptor (HIA) Device Module - provides a generic interface, in combination with AN/UYK-43 executive software, to embedded AN/UYK-43 HIA devices, including the 68030-based Time Critical Subfunction (TCS) Coprocessor.

i. PC Debug - microprocessor-based program providing remote manipulation of the AN/UYK-43 Display Control Unit (DCU) and access to AN/UYK-43 software to efficiently debug and monitor programs real time.

j. Resident Debug - basic software checkout functions completely resident in the AN/UYK-43.

k. Utility Package (UPAK) - off-line tools for initial system building and for loading, maintenance, and preliminary checkout of software.

l. Data Extraction - on-line tool for capturing specified system operational data at specified times and in a well-defined format for subsequent data reduction.

m. On-line Data Extraction - tool for taking data captured by the Data Extraction module, selecting and correlating specified items, and producing formatted printer reports. Customers may use this tool in an operational environment on board ship or in a stand-alone mode under 43RSS in the laboratory.

4.2.8.1 Operating System Service Classes

4.2.8.1.1 General Requirements

Through the MTASS and SHARE program generation systems, 43RSS supports software written in the CMS-2 and MACRO Assembly languages.

4.2.8.1.2 Architecture Dependent Interfaces

Hardware - 43RSS is built to execute on and make the best use of the AN/UYK-43 computer. 43RSS is an outgrowth of SDEX/7 and the ModX2 Common Program which ran on and utilized the AN/UYK-7 computer.

Software - SDEX/43 supports modularly structured software consisting of various module entrances (e.g., successor, message, time dependent, background). SDEX/7 also supported this kind of software. However, SDEX/43 has also captured the functionality of the Multiprocessor Computer Executive Program (MCEP) and now also supports software with a task structure. In contrast to modules, these tasks have a single thread of execution. They release control to wait on Input/Output (I/O), on a timed event, or for signalling by another task.

4.2.8.1.3 Capability and Security Interfaces

43RSS is not rated for security at this time. An analysis of 43RSS has been done specifying the changes that would be necessary for C2 and B1 ratings.

43RSS does not provide a nuclear safety option at this time. Should a system require nuclear safety in the future, enhancements to 43RSS could provide capabilities similar to those provided by SDEX/44.

4.2.8.1.4 Data Interchange Interfaces

Common Peripheral converts and formats data for output to peripheral devices.

4.2.8.1.5 Event and Error Management Interfaces

Event Management

Both File Handler and Common Peripheral provide a semaphore option to suspend requesting applications software until I/O processing completes.

Error Management

Error processing in FAM is table driven where the tables specify recovery options selected by the system designer for the various classes and subclasses of errors.

43RSS error recovery routines pass control to 43RSS system initialization processing. Eventually, control passes to the 43RSS FAM component which can do any of the following:

- Load/delete application software by individual programs or by configurations of programs during real time operations. FAM can do this automatically on FAM initialization or through commands from the system operator.
- Process hardware and software error interrupts which include isolating hardware module failures and establishing the optimal recovery option.
- Display the current operational status of AN/UYK-43 hardware modules and peripheral equipment.
- Roll out critical data during system execution and allow for reloading the critical data upon reinitializing the system.
- Display hardware and software errors and also display the recommended diagnostics associated with each error.

4.2.8.1.6 File Interfaces

The File Handler and Common Peripheral components provide the I/O functions in 43RSS. The File Handler provides a file system for magnetic disk and magnetic tape. The file system supports two types of files: block-level and record-level files. Block-level I/O maps logical-sized blocks onto physical cylinders, tracks, sectors, etc., of the target mass storage device. File Handler passes data directly to the user specified buffer area from the mass storage device. Record-level I/O provides an indexed logical record file structure for a file. Users obtain data through the use of an indexed-random access or sequential indexed order access. Each record within a file has a user-defined fixed length. Blocks of data are passed from the mass storage device to the File Handler system buffers where the record of data is passed to the user.

4.2.8.1.7 Generalized I/O Interfaces

SDEX/43 provides the means for user modules to initiate and control I/O operations. This includes initiating I/O in response to a hardware interrupt, registering to receive CP control following an I/O interrupt, enabling or disabling external interrupts on an I/O channel, or calling SDEX/43 to initiate an I/O transfer.

The Common Peripheral (CP) component manages user software access to system peripheral devices. This access controls I/O, converts and formats data, coordinates multiple peripheral devices

within a category and monitors the integrity of communications with the peripheral devices. CP manages queueing and dequeueing on its I/O channels and notification of the user when I/O has completed.

Various kinds of information and control are available to a 43RSS system operator. PC Debug and Resident Debug provide an operator interface either through a PC or through a designated system control console. In the case of PC Debug, the operator uses menus to access functions and may use a Job Control Language to generate command runstreams. UPAK provides commands to display and modify main memory and control peripheral devices. Data Extraction and On-line Data Reduction allow the operator to specify parameters and control execution through a system console. Through the designated system console, the operator can use the Common Peripheral component to specify I/O channel assignments, control magnetic tapes, and communicate messages to user software. The operator can use the Common System component to set peripheral devices operable or inoperable. The operator can use DMR to get information about how main memory is mapped. The operator can use FAM to control I/O channel assignments, control the system software configuration, control hardware diagnostics, and roll system data in and out from disk.

43RSS provides general purpose operator notification and recovery processing if an error interrupt occurs. In most cases, on-line operator help functions are available.

4.2.8.1.8 Networks and Communications

43RSS has participated in the SAFENET committees. As an item in its PI program, 43RSS has a plan to implement a SAFENET User Agent interface for its users. 43RSS implemented the interface to the TCS Coprocessor using a message passing scheme patterned on the SAFENET User Agent interface protocol.

4.2.8.1.9 Process Management Interfaces

43RSS arbitrates the allocation of AN/UYK-43 Central Processor Unit (CPU) resources to user software according to the priority, message passing, periodic, background, and immediate processing requirements of real time, tactical systems.

43RSS provides for the creation, initiation, and normal and forced termination of user software. DMR establishes software in the system as it loads the software. Executive Service Requests (ESRs) to SDEX/43 result in the necessary housekeeping to initiate the software and to terminate the software.

SDEX/43 schedules and dispatches software according to the following type of entrances:

a. Successor - for high priority real time processing
b. Message - for internal communication between modules
c. Time dependent - for periodically repeated processing
d. Background - for processing not executed on a real time basis
e. I/O interrupt - for processing executed as a result of I/O

controller (IOC) channel monitor interrupts

f. Time critical - for processing executed at a specific time interval as controlled by the IOC monitor clock.

SDEX/43 queues the first four entrance types. Successor and time dependent entrances may be time sliced. Background entrances are always time sliced. SDEX/43 does not queue I/O interrupt and time critical entrances; they receive immediate processor control when they occur.

Currently, 43RSS provides intercomputer communication and control for an I/O channel. This service provides for message transfer, including determining if the receiver is local or remote and directing the message accordingly; clock synchronization; and maintaining status of the remote computer.

43RSS has participated in the SAFENET committees. As an item in its Product Improvement (PI) program, 43RSS has a plan to implement a SAFENET User Agent interface for its users. 43RSS has implemented the interface to the TCS Coprocessor using a message passing scheme patterned on the SAFENET User Agent interface protocol.

4.2.8.1.10 Project Support Environment Interfaces

43RSS is coded and maintained using the U.S. Navy standard Machine Transferable Support Software for the U.S. Navy Standard Computers (MTASS) and SHARE program generation packages. System build is thus facilitated for software developed on MTASS or CMS-2Y systems. The MTASS host computers include the Unisys 1100 series; the IBM 360, 370, and 4341 series; and the DEC VAX-11/780. SHARE is hosted on the AN/UYK-7 and AN/UYK-43 computers.

4.2.8.1.11 Reliability, Adaptability, and Maintainability Interfaces

Reliability

43RSS, through the SDEX/43, FIM, and FAM components, extends the fault tolerant capabilities of the AN/UYK-43 to provide a comprehensive AN/UYK-43 fault tolerant system. SDEX/43 provides registration of a module for error responsibility. In a system making use of the full fault tolerant capabilities of the AN/UYK-43 and 43RSS, the error module is FAM.

SDEX/43 provides module access to CPU and IOC confidence testing and to the AN/UYK-43 firmware FTSRM for modifying FTSRM parameters, requesting a hardware module resource check, requesting on-line hardware diagnostics, entering interrupt sanity processing, and requesting a system reload. SDEX/43 also provides for reading the AN/UYK-43 execution address history table, using the CPU and IOC hardware breakpoints, and using software breakpoints.

FIM provides for initialization of FTSRM parameters and in its regular processing makes use of FTSRM functions: updating hardware resources status, executing on-line diagnostics and processing errors. FIM also performs, on a cyclic basis and under

system direction, CPU and IOC confidence testing to improve system reliability and fault detection capabilities. It makes diagnostic recommendations based on modules that FTSRM has found suspect and on interrupt status codes to FAM as well as supplying FAM and FTSRM parameters and error data. FAM manages program load and reconfiguration, rolls in and out critical data for orderly module start-up, and makes diagnostics test decisions.

Adaptability

Users tailor 43RSS to their particular system requirements through ordering parameters. These ordering parameters may specify compile-time options. Instructions and data for an option are not included in the compiled version of 43RSS for a user if the user did not select that option. Other ordering parameters define system parameters such as table sizes and the hardware configuration. If a system requirement is not satisfied by an available option, customers can request Engineering Change Proposals (ECPs) to 43RSS through the U.S. Navy.

43RSS has an active PI program which is designed to keep pace with the AN/UYK-43 hardware PI. It, therefore, includes items such as:

- Expanded Memory Reach - provides access to more than 8000 base registers being added to the AN/UYK-43.
- Time Critical Subfunction (TCS) Coprocessor - provides an interface to the AN/UYK-43 embedded 68030-based coprocessor.
- Local Area Network (LAN) - provides an interface to a SAFENET compatible LAN.
- Embedded Memory Subsystem - provides access to AN/UYK-43 embedded mass memory.

43RSS PI program is also designed to keep pace with developing user system needs such as the ability to run ALS/N Ada/L software with existing CMS-2 software in a mixed language system.

Maintainability

The Data Extraction component gathers data during such activities as system integration and certification, shipboard dock-side tests, sea trials, operational patrols, and training. The Data Extraction module captures system operational data and outputs it in a well defined format for subsequent data reduction.

4.2.8.1.12 Resource Management Interfaces

Memory Management

43RSS provides for the dynamic loading and management of AN/UYK-43 main memory (currently up to 20M 32-bit words) and the control of the user software view of memory through hardware base registers.

The DMR component is responsible for the required software loading and the management of the AN/UYK-43 memory. DMR assigns memory for loading of required program segments, for loading program segments or configurations of programs, and for memory segments requested by user software. It provides for loading

instructions and data in different memory banks for more efficient memory access, for loading in semi-conductor memory or core memory, and for loading in or not in a memory back containing the 43RSS executive software. Users can send requests to DMR to acquire/release blocks of memory and to load/delete module segments.

DMR uses one of three memory allocation algorithms selectable by a user ordering parameter: first-fit consolidates memory usage in the lowest available memory units, best-fit reduces memory fragmentation, or worst-fit reduces memory access conflicts.

DMR maintains the locations of all segments in memory and consolidates and identifies all unassigned memory.

The original AN/UYK-43 computer provided eight task state base registers for a 512K (8 registers of 64K words) view of memory for task state applications software. SDEX/43 aids user software in using this view by loading the base and memory protection registers according to the user software direction. The AN/UYK-43 is currently being enhanced to provide 8176 new base registers and 43RSS is likewise being enhanced to provide its users access to these base registers.

Device Management

Common Peripheral contains all the device drivers for the 43RSS system. Although the File Handler manages the files on magnetic disk and magnetic tape, Common Peripheral performs the actual I/O. Common Peripheral currently supports magnetic tape, magnetic disk, keyboard/print/display, and line printer devices.

4.2.8.1.13 Synchronization and Scheduling Interfaces

SDEX/43 provides for the use of software semaphores. Semaphores allow for module entrance synchronization and rendezvous, as well as controlled access to system code and data.

Both File Handler and Common Peripheral provide a semaphore option to suspend requesting applications software until I/O processing completes.

The scheduling services provided by 43RSS are discussed in the Process Management section above.

4.2.8.1.14 System Initialization and Reinitialization Interfaces

43RSS provides system initialization, and system reload and restart with or without saved data.

The 43RSS DMR component loads and initializes the 43RSS operational system. Initially, DMR receives control from the firmware FTSRM bootstrap program, establishes and initializes the various parts of the AN/UYK-43 hardware and firmware as required, and initializes the 43RSS executive and base program modules. DMR then performs the required software loading, provides memory management, and responds to operator requests for load and resource information printouts.

In the case of both boot load and restart processing, DMR receives control from the FTSRM bootstrap routines, initializes the

executive, loads the required software, and activates it by sending each loaded module a preset message. The preset message allows the loaded modules to initialize and begin normal processing. If the operator has specified a restart at the computer control panel and rolled out data is available, FAM will perform restart processing. FAM will load the applications modules it is responsible for, roll in saved data, and send the loaded modules restart preset messages.

4.2.8.1.15 Time Interfaces

SDEX/43 provides user access to the various AN/UYK-43 hardware clocks and the software real time clock. In general, read access is available. Write access is also available for the AN/UYK-43 calendar clock and users can change the tick rate for the AN/UYK-43 real time clock not being used for system timing.

4.2.8.1.16 Ada Language Support Interfaces

43RSS is currently pursuing providing support for Ada software developed under ALS/N Ada/L to run in a mixed CMS-2 and Ada software system.

4.2.8.2 Additional Characteristics

4.2.8.2.1 Proprietary or Open

43RSS is sponsored by NAVSEA PMS 412.

4.2.8.2.2 Qualification as a Standard

43RSS is a U.S. Navy standard operating system for the AN/UYK-43.

4.2.8.2.3 Platform Flexibility

43RSS is only targeted for the AN/UYK-43 computer.

4.2.8.2.4 Application Domain

The following projects use some or all of the components of 43RSS:

- Aegis Cruiser
- Trident Submarine
- (CVN) Cruiser
- AN/UYK-43 Operational Test System (OTS)
- CCS MK II
- SSN Submarine (SOS/43)
- AN/UYK-43 Relational Data Base Management System (43RDBMS)
- LHD-1 Amphibious Ship
- LHD-2 Amphibious Ship

- CG-34 Cruiser
- Runtime Support Software/Rehosted Simulation Control Program (RSS/RSCP)
- German Navy Frigate F-123 ship.

4.2.8.2.5 Testability and Performance Evaluation Mechanism

43RSS provides various tools for both software debug and system monitoring. SDEX/43 provides mechanisms such as user selectable recording of various events, breakpoints, and access to the AN/UYK-43 jump and interrupt return history. The PC Debug component provides for control of the AN/UYK-43 Display Control Unit as well as mechanisms similar to those provided by SDEX/43 and other dumps and system monitoring information in operator oriented formats at a personal computer. Resident Debug provides for system monitoring and test by an operator through the system keyboard/display. UPAK provides for off-line dumping of AN/UYK-43 registers and memory locations. Data Extraction and On-line Data Reduction provide for the extracting and on-line reducing of operational system data.

4.2.8.3 References

[KRUEM89]

4.2.9 iRMX Survey Summary

The Distributed iRMXO Operating System is the newest member of the iRMX family of operating systems which has served as the base operating system for thousands of real-time applications. The Distributed iRMX Operating System has been designed to extend the capabilities of the traditional iRMX operating systems to serve as the basis for the construction of real-time multicomputer systems consisting of multiple single board microcomputers interconnected by a high-speed, message oriented bus. This operating system provides both functional distribution and transparent multiprocessing. Functional distribution is realized through the assignment of various tasks (e.g., program development, file management, real-time application processing) to separate components of the distributed system. The program development function is provided by a separate single board microcomputer running the UNIX1 Operating System. The real-time application processing component is implemented as a distributed system with the components of the system consisting of single board microcomputers based on the Intel 386 microprocessor. This object oriented operating system is implemented on each host of the real-time system comprising the real-time and file management components. Each instance of the Distributed iRMX Operating System cooperates with other instances residing on other hosts in the real-time system to unify the management of the system's resources. The result is a transparent, multiprocessing environment that encourages the construction of distributed, real-time applications.

The target for the Distributed iRMX Operating System is a multiple computer system consisting of multiple 386 CPU-based single board microcomputers interconnected via the MULTIBUSO II bus. The effective bandwidth of the MULTIBUS II bus represents a significant improvement over traditional LAN-based systems. The individual copies of the Distributed iRMX Operating System running on each of the single board microcomputers cooperate to provide to the user a unified collection of resources. The user is presented with an environment for constructing distributed real-time applications that contains programming constructs familiar to a user of a standard uniprocessor real-time system.

iRMX is a registered trademark of Intel Corp.

1UNIX is a registered trademark of AT&T Bell Laboratories.

386 is a trademark of Intel Corp.

MULTIBUS is a registered trademark of Intel Corp.

iRMK is a trademark of Intel Corp.

The loosely coupled nature of the target hardware had a significant impact on the design of the Distributed iRMX Operating System. We realized early the advantages of functional distribution and as mentioned earlier decided to make that a fundamental part of our design. In particular, we chose to use the standard UNIX Operating System to serve as the basis for the program development component, and let the Distributed iRMX Operating System provide the environment for executing the real-time applications including real-time computation and I/O processing. Thus, users are provided with a familiar environment for program development as well as data input and data analysis while a physically separate subsystem is provided to support the much different demands of real-time processing. The ability to share files combined with transport level communication support combines the functionally separated components into a single system.

The Distributed iRMX Operating System is a hierarchically designed operating system consisting of the following layers:

* The iRMX Kernel: the base of the full operating system is the iRMX Kernel, which is also sold as a stand-alone product. The Kernel provides typical kernel facilities including task management, interrupt management, time management, and basic device management. In addition, the Kernel provides both data link and transport access to the MULTIBUS II bus. The Kernel is designed to be flexible allowing users to easily extend its capability. This has proven useful to users of the stand-alone kernel product as well as the developers of the full operating system. A unique capability of the Distributed iRMX Operating System is the direct access to the facilities of the Kernel made available to users of the operating system. In most other systems, this access is limited to the modules of the operating system.
* The Nucleus: the Nucleus runs on top of the Kernel providing

a greater level of protection for the basicservices offered by the Kernel. In addition, it is at this level where the notion of distribution is introduced. Examples include support for remote job creation and intra and interprocessor mailbox communication.

* Network Transport Communication: in contrast to the MULTIBUS II Transport protocol implemented by the iRMX Kernel which is intended to provide a transport facility optimized for the MULTIBUS II bus, the full operating system also offers a generic network transport service called the OSI Transport Service (OTS). This facility views the MULTIBUS II bus as a network and includes a gateway to allow the MULTIBUS II system to be connected to other networks (e.g., a traditional LAN).

* I/O System: The I/O System is functionally partitioned into client and server portions to provide full file sharing services to any host in the system. Full flexibility is provided for multiple clients, multiple servers, and to support both intelligent and non-intelligent I/O controllers.

4.2.9.1 Operating System Service Classes

4.2.9.1.1 General Requirements

The Distributed iRMX Operating System is an object oriented operating system. Interfaces are provided for the creation, deletion, and manipulation of specific system objects. In addition, a set of generic object operations are also provided (e.g., get object type, lock/unlock an object against deletion). Users are provided interfaces to create new object types. In addition, there is support for users creating managers for the new object types including associating additional data with various objects in order to customize the objects and establishing handlers to be invoked upon creation and deletion of jobs, the operating environment for tasks.

Two levels of interface are available to users of the Distributed iRMX Operating System. The higher level is the operating system level described in the preceding paragraph. The lower level is the kernel level. An implementation of the kernel level interface is also currently available in a separate kernel product, the iRMX Kernel. As with the operating system level interfaces, the kernel level interfaces are object oriented. They differ somewhat in that they provide somewhat more control to the user than the operating system level interfaces.

The interfaces are not tied to any particular architecture or implementation. They all follow a standard format that includes the following parameter ordering:

(1) token: used to identify the object being manipulated. This is a 32-bit entity that is not interpreted directly by the user. It is created by the operating system when the object is created, passed to the user, and used by the user in further calls to object manipulation interfaces.

(2) miscellaneous parameters: various parameters specific to

the interface.
(3) flags: a bit map identifying various applications of the interface.A common flag bit is the scope bit that indicates whether the operation is to be limited to the local job (local environment) or applies to the entire distributed system.
(4) status: exception status returned to the user.
The current implementation provides interfaces for C and PL/M applications. The same interface format is used for both languages. In fact, a common reference manual is used to support both languages. Users also have access to the interfaces provided by the underlying kernel, the iRMX Kernel. The iRMX Kernel also provides interfaces to Ada and Fortran applications. Again, there exist no differences in the interfaces provided for the various languages.

4.2.9.1.2 Architecture Dependent Interfaces

As mentioned above, the Distributed iRMX Operating System is an object oriented system that provides the user with the ability to add new objects and define new operations for manipulating these objects. The support for distribution can be made transparent allowing users to view the system as a cohesive set of resources. Partitioning is along logical boundaries, the job or operating environment, rather than physical processor boundaries. The Distributed iRMX Operating System is implemented on top of the iRMX Kernel which provides a virtual machine environment to the operating system. The iRMX Kernel has already been implemented on two vastly different architectures, the 386 and the 960 (a RISC architecture).
Communication interfaces are provided at a number of levels. This includes a simple mailbox communication mechanism, a request response communication mechanism, and a network transport facility. The mailbox communication interfaces are rather simple. Each mailbox is an object that is referenced by a token. The send interface simply takes a token, a pointer to a message, and the length of the message. The message is either copied to a task already waiting at the mailbox or copied into the mailbox if no tasks are waiting. The request-response communication interfaces are somewhat more general and thus more powerful. Users have the ability to associate a transaction id with each message, thus enabling users to wait for response messages to specific request messages. In addition, users can separate a message into separate control and data parts. These interfaces allow a user to send a message consisting of a collection of disjoint fragments chained together. In addition, interfaces are provided to support a fragmentation protocol in which one task can either send or receive a complete message while the task on the other end deals with the transmission as a series of transmissions of message fragments. Finally, the Distributed iRMX Operating System provides network transport communication interfaces.
While the current implementation of the Distributed iRMX Operating System doesn't provide any support for high performance,

shared memory multiprocessing, it should be easily extendible to provide this support. In general, we believe very little additional support will be required. For example, the existing shared memory synchronization facilities (e.g., semaphores) need not be modified to provide support in a shared memory environment. Only the implementation will need modification.

4.2.9.1.3 Capability and Security Interfaces

The Distributed iRMX Operating System uses a privilege ring protection mechanism. Associated with each object is a privilege level which limits access to the object. Associated with each task is a privilege level. A task's privilege level must equal or exceed that of an object in order to manipulate the object. When an object is created it is assigned the same privilege level as the task that is performing the creation. The object's privilege level can by dynamically adjusted, but it cannot be increased beyond that of the privilege level of the task requesting the change. The privilege mechanism is also used to limit access to certain interfaces. This limit is defined during the operating system configuration process.

4.2.9.1.4 Data Interchange Interfaces

There is no direct support in the Distributed iRMX Operating System interface for data interchange services.

4.2.9.1.5 Event and Error Management Interfaces

The Distributed iRMX Operating System provides a full range of interrupt management and exception management facilities. As a real-time operating system, the timeliness and predictability of delivery of interrupts is of utmost concern. All interfaces that have the potential for blocking for unpredictable periods of time include a timeout parameter that specifies time in milliseconds. Interfaces are provided for selectively disabling/enabling interrupts. In addition, the priority of interrupts is tied to the task priority scheme in such a fashion that a task running at a particular priority will implicitly disable interrupts of equal and lesser priority. This provides users with closer control over the processing of internal work while dealing with asynchronous external events. Interrupts can be handled in one of the following three methods:

(1) Interrupt Handler: this is a procedure that is invoked in the context of the currently running task. While this is the fastest method for dealing with interrupts, it also has a couple of limitations. First, during the handling of the inter- rupt by the interrupt handler, all interrupts are disabled. Second, it is not safe to execute most of the other system calls during processing by the interrupt handler.

(2) Interrupt Task: this is a separate task assigned to processing the interrupt. A task switch to the interrupt task

occurs upon reception of the interrupt signal. Thus, the interrupt is processed in the context of a special dedicated task. This is slower than the interrupt handler facility but eliminates the two limitations of the handler technique described in the preceding item.

(3) Interrupt Handler/Task Pair: this combines the best of both of the two preceding techniques. Both an interrupt handler and an interrupt task are assigned to handle an interrupt. When an interrupt is received, the interrupt handler will be invoked in the context of the currently running task. While this handler executes, the limitations identified previously for the interrupt handler apply. If the processing required for the interrupt is sufficiently small, all work can be handled by the handler alone. If more extensive processing is required, a special interface is provided to allow the interrupt handler to signal the interrupt task of the arrival of the interrupt. The handler then exits and a task switch to the interrupt task occurs. The interrupt task looks like any other task except that it is designed around a big loop where it waits for an interrupt signal and then processes that interrupt. The full set of system calls are available to the interrupt task.

As mentioned at the start of this section, all blocking interfaces contain a timeout parameter. In addition, there is a special alarm facility available in the kernel level interfaces. This facility allows for the definition of a single shot or repeat alarm that will result in the invocation of a handler procedure in the event the specified timeout value is reached.

The Distributed iRMX Operating System also defines a flexible exception management facility. Users can specify whether they wish to handle exceptions from system calls in-line or with a handler. They can make their selection for each privilege ring as well as on a per job or per task basis. Users can also raise an exception with the aid of a special interface.

In addition to handling the traditional exception management needs, a special facility for dealing with host failure in a distributed environment is defined. From an implementation point of view, the Distributed iRMX Operating System provides a software watchdog timer facility to monitor the liveness of the various hosts in the system. Host failure and reset situations are detected by dedicated software watchdog timer tasks executing on each host. When each host detects the failure or reset condition, a reconfiguration message indicating the effected host and a failure or reset indication is sent to the reconfiguration mailboxes that have been registered with the software watchdog timer. Those operating system type managers that need to perform recovery actions will dedicate a recovery task to this activity. This task will create a mailbox and register it with the software watchdog timer. An operating system interface has been defined to also allow user applications to register mailboxes with the software watchdog timer in order to allow them to participate in the recovery process.

To further isolate problems caused by remote host failure and

reset, the tokens used to access objects contain incarnation information used to distinguish a token created on a failed host from that created on a new incarnation of that host. As mentioned earlier, the token is not directly interpreted by the user. The presence of this field in the token, though, allows the operating system implementation to detect references to objects on a failed host and reject these requests.

4.2.9.1.6 File Interfaces

The Distributed iRMX Operating System includes a full featured distributed I/O system that implements basically a Berkeley Unix file system. Access control and the ability to read/write an arbitrary number of bytes of data in an arbitrary location in a file is provided as in the Berkeley Unix system. Locking at the byte level is supported. The I/O interfaces can either be executed synchronously or asynchronously. The distributed architecture is used for defining the I/O system. In particular, the I/O system is viewed as consisting of two types of components, front-ends and file servers. The file management interfaces are implemented by the front-ends who send the appropriate messages to the file servers which directly manipulate the media.

4.2.9.1.7 Generalized I/O Interfaces

A procedural protocol between the file server and the device driver has been defined for the Distributed iRMX Operating System. This enables user configurable device drivers to be supported, thus allowing access to a wide variety of devices with interfaces at a file level of abstraction.

4.2.9.1.8 Network and Communications Interfaces

The Distributed iRMX Operating System provides network transport communication interfaces that are functionally equivalent to Unix TLI but have object orientation to make them similar to the other Distributed iRMX interfaces. In addition, interfaces supporting a name service allowing network addresses to be cataloged and retrieved by symbolic name is provided.

4.2.9.1.9 Process Management Interfaces

The Distributed iRMX Operating System defines the concept of a job as the operating environment for one or more tasks. A job is somewhat like a Unix process and a task is somewhat like a Unix thread or lightweight process. It defines an environment for the creation of objects including tasks. The memory used in the creation of objects by tasks operating within a job is obtained from a common memory pool allocated to the job. Objects created with a local scope specification can only be accessed by tasks operating within the job. Objects created with global scope can be accessed from outside of the job.

Interfaces are provided for the creation and deletion of jobs. Jobs can either be created as independent (first level) or dependent jobs. Dependent jobs have the notion of a parent. The Distributed iRMX Operating System defines the notion of a job tree representing the complete set of parent-child relationships for all jobs in the distributed system. Independent jobs are defined to be the children of a fictional root job. The parent-child relationships are maintained across processor boundaries. Parent-child relationships are particularly important in terms of actions that occur in the event of explicit job deletion or implicit job deletion resulting from host failure. In either case, the operating system enforces the principle of not allowing orphans. Thus, the entire subtree rooted at a deleted job will automatically be deleted by the operating system. Jobs can also be suspended/resumed. In this case all tasks within a job are suspended/resumed.

Operations on tasks include create, delete, suspend, and resume. Tasks can be created in either a ready or suspended state. A sleep primitive is also available to allow a task to perform a timed wait. The time parameter to the sleep primitive is defined in units of milliseconds.

The following three basic interprocess communication facilities are provided:

(1) Semaphores: basic counting semaphores are provided. The wait and signal interfaces, called receive_unit and send_unit, allow a user to receive/send multiple units with a single call.

(2) Mailboxes: mailboxes provide a simple message exchange facility that allows communication between tasks residing on different hosts.

(3) Ports: ports provide a more full featured message communication facility that supports request/response communication, provides separation of messages into control and data parts, allows the message to consist of multiple fragments chained together, and provides the ability to send or receive a message in multiple fragments while the other communicating partner deals with the message as a single unit.

4.2.9.1.10 Project Support Environment Interfaces

The Distributed iRMX Operating System is designed to operate with a symbolic static and tasking debugger that has been customized to be aware of various operating system objects. The debugger interface defines the ability to disassemble code, examine registers, read/write memory, single step, single step with stepping through procedures, setting/clearing code and data breakpoints. The breakpoint facility has been enhanced to support the Distributed iRMX Operating System environment by allowing the user to restrict the scope of a breakpoint to a job or a task. In addition to these basic debug facilities, a set of operating system awareness extensions have been provided to allow display of operating system objects and key system data structures (e.g., job

tree, ready queue on a particular processor). The debugger interface is designed to allow the user to debug a distributed application with little or no regard to knowledge of the location where the application is running.

4.2.9.1.11 Reliability, Adaptability, and Maintainability Interfaces

The Distributed iRMX Operating system allows users to configure handlers for various hardware faults. In addition to handling traditional uniprocessor faults, support is provided for detecting and recovering from host failure. This capability is provided by the software watchdog timer facility described in the Event and Error Management Interfaces section.

4.2.9.1.12 Resource Management Interface

When the Distributed iRMX Operating System is configured, users can define the memory to be assigned to various logical memory pools used for allocation of objects at runtime. The 386-based implementation of the Distributed iRMX Operating System utilizes the hardware memory protection facilities. This includes bounds checking, access rights checking, and privilege ring protection.

4.2.9.1.13 Synchronization/Scheduling Interface

The Distributed iRMX Operating System uses a preemptive, priority based scheduling mechanism. Users can control the scheduling mechanism at a variety of levels. In particular at the time the operating system is configured, users can control the use of time slicing through the establishment of the value of the real-time fence. This corresponds to a task priority value and indicates that tasks of equal or higher priority are not time sliced. In addition, users can dynamically modify task priorities as well as the timeslice interval for a task. At the kernel level, users can dynamically associate any number of handlers to be invoked upon task creation, task deletion, task switch, and task priority change. These lower level interfaces provide a powerful mechanism for implementing a wide variety of scheduling policies.

In terms of task synchronization, a semaphore mechanism discussed in Section 5.2.8.1.9 is provided.

4.2.9.1.14 System Initialization and Reinitialization Interfaces

The 386, MULTIBUS II implementation of the Distributed iRMX Operating System provides interfaces for accessing the interconnect space register set of various hosts in the system. Interconnect space registers are used to provide an area for data exchange between various hardware, firmware, and software components with emphasis placed on the needs for such communication during system initialization.

4.2.9.1.15 Time Services Interface

The Distributed iRMX Operating System provides interfaces to read and set a clock on each host. In addition at the kernel level, interfaces are provided for programming a Programmable Interval Timer (PIT) device. As mentioned in Section 5.2.8.1.5, an alarm facility is defined at the kernel level.

4.2.9.1.16 Ada Language Support Interface

No special Ada support is currently provided. As mentioned in Section 5.2.8.1.1, an Ada binding is currently provided for the iRMX Kernel interfaces. No interface changes exist between those provided for Ada, C, PL/M, and Fortran.

4.2.9.2 Additional Characteristics

4.2.9.3 References

4.2.10 Mach Survey Summary

The Mach project was initiated at Carnegie Mellon University (CMU) in 1984 as the operating system effort of DARPA's Strategic Computing Initiative (SCI). Mach was envisioned as an operating system that would (1) provide a uniform (UNIX-compatible) software base across the architectures existing at the time, as well as the new advanced architectures being developed as part of the SCI, and (2) support the interconnection of these architectures into distributed computing environments. Its goals may be elaborated as follows:

o Mach was designed to extend UNIX functionality to multiprocessor architectures, ranging from (1) uniform access, shared memory multiprocessors (UMA, for Uniform Memory Architecture) (e.g., Encore Multimax, Sequent Balance), to (2) differential access, shared memory multiprocessors (NUMA, for non-UMA) (e.g., BBN Butterfly, IBM RP3), to (3) multicomputer architectures (NORMA, for No Remote Memory Access Architecture) (e.g., hypercube).

o Mach was designed to extend UNIX functionality to large memory architectures.

o Mach was designed to extend UNIX functionality to distributed computing environments, in which diverse architectures (i.e., uniprocessors, multiprocessors) interconnected by high speed networks support distributed applications.

o To take advantage of the vast supply of UNIX-based software,

Mach was designed to offer (and continues to offer) UNIX compatibility (specifically, binary compatibility with 4.3 BSD).

Although Mach offers UNIX compatibility, it is not intended to be bound to UNIX. The current, evolved vision is for the Mach distributed operating system to be based on a minimal kernel upon which multiple operating system environments can be built. At this point, the kernelization is not complete, and some UNIX functionality is still embedded in Mach kernel code. When the kernelization is complete, it will be possible to emulate operating system environments other than UNIX 4.3 BSD on top of the Mach kernel.

4.2.10.1 Operating System Service Classes

4.2.10.1.1 General Requirements

An interface specification language, MIG (Mach Interface Generator), has been developed for Mach. MIG generates C or Common Lisp RPC stubs.

4.2.10.1.2 Architecture Dependent Interfaces

No explicit features

4.2.10.1.3 Capability and Security Interfaces

Naming and Protection

As noted in the IPC section, the Mach kernel uses capabilities, in the form of ports, for naming and protection on a single system.

The network message servers extend the protection to the network environment, by implementing mechanisms to protect both the messages sent over the network to network ports and the network port capabilities.

Security - Trusted Mach

The Trusted Mach project is a DARPA-sponsored research effort of Trusted Information Systems, Inc. The goal is to build a version of Mach - Trusted Mach - that meets the B3 level of protection as specified in the National Computer Security Center (NCSC) Trusted Computer System Evaluation Criteria (TCSEC), the so-called "Orange Book" [TCSEC 85].

The project adopts the idea of "incremental reference monitors." At the lowest level is the Trusted Mach Kernel. At the intermediate level is the reference monitor composed of the kernel and a trusted name server. At the highest level is the reference monitor composed of the kernel, a trusted name server, and other trusted servers. Thus far, work has concentrated on the kernel level of a single machine. Mach's ports are serving as the protected objects in Trusted Mach; its tasks (through their threads, which are the active entities) are serving as the subjects. Extensions are being developed to meet the TCSEC

requirements for both discretionary and mandatory protection.

At this time, the Trusted Mach project is utilizing a Spring 1988 version of Mach. Since this version is not kernelized, the effort cannot yield a trusted operating system. The unkernelized version of Mach is serving as a platform for research into multilevel security, not as a base upon which to build a trusted system. The development of a trusted version is tied to the completion of Mach kernelization.

Security - Strongbox

Strongbox is built on top of Camelot and Mach. It is based upon the new concept of "self-securing" programs, i.e., programs that can run securely on distributed operating systems (such as Mach) that provide only minimal security facilities.

Two key algorithms implemented by Strongbox are zero knowledge authentication and fingerprinting.

It should be noted that Strongbox is (currently) concerned with the security issues that arise from protecting the privacy of data and ensuring the integrity of data from alteration; security issues of denial of service, covert channel analysis, and traffic analysis of message patterns have not been considered, although they could be.

4.2.10.1.4 Data Interchange Interfaces

Matchmaker is an interface specification language for use with existing programming languages. Differences in type representation by various programming languages within each machine are handled by Matchmaker. Data representation issues across machine boundaries are handled through message server processes. Byte reordering and machine specific conversions are performed by the message servers with the responsibility for conversion always resting with the receiving host.

4.2.10.1.5 Event and Error Management Interfaces

Mach utilizes message passing for the invocation of exception handlers. When a thread raises an exception, a message is sent to its thread exception port to notify its error handler, which executes in a separate thread. If no handler exists or the handler fails to recover the exception, the message is forwarded to the exception port of the task in which the exception-incurring thread exists.

A debugger can intercept unhandled exceptions for all threads in a task by attaching itself to the task exception port. This enables a debugger to coexist with error handlers, in that the debugger is aware only of those exceptions not handled by an error handler.

4.2.10.1.6 File Interfaces

The current vision is for Mach to be based on a minimal

kernel upon which multiple file systems can be built. At this point, the kernelization is not complete, and the UNIX file system (4.3BSD)functionality is still embedded in Mach kernel code.

4.2.10.1.7 Generalized I/O Interfaces

Currently, the UNIX I/O interface is used with low-level device drivers residing in the Mach kernel. Virtual-memory based file-mapping replaces buffer management in the standard I/O libraries.

4.2.10.1.8 Networks and Communications

Interprocess Communication

Mach interprocess communication (IPC) is based on the port and message abstractions. Ports are the reference objects in Mach, and, as such, are viewed as playing the same role as capabilities in an object-oriented system. Objects such as tasks, threads, and memory objects are represented as ports, and operations on these objects are performed by sending messages to the ports that represent them. Only tasks with send rights to a port can send messages to it, and only the (single) task with receive rights to a port can receive messages from it.

Messages can be sent and received synchronously (as in Remote Procedure Calls (RPCs)) or asynchronously. They can contain capabilities. In fact, the only way for a task to acquire a capability is to receive it in a message.

In Mach, the kernel itself implements local IPC only. However, a user-state task, called the network message server, transparently extends IPC into a network environment. This task maintains mappings of local "proxy" ports to global "network" ports. It forwards messages using network protocols of its choice.

Naming

The Netmsgserver passes all the Mach IPC message between machines. It also provides network wide port register and lookup functions.

The Environment Manager can register or look up ports or named strings but does not communicate with other Environment Managers.

4.2.10.1.9 Process Management Interfaces

Basic Abstractions

Mach divides the process abstraction into two orthogonal abstractions: the task and the thread. A task is a collection of system resources. These include a virtual address space and a set of port rights. The thread is the basic unit of computation; it is the specification of an execution state within a task. Mach allows multiple threads to execute within a single task.

Operations on tasks and threads are invoked by sending a message to a port representing the task or thread. Threads maybe be created, destroyed, suspended, and resumed.

Tasks are related to one another in a tree structure by task-creation operations. Regions of virtual memory may be marked for future child tasks as either inheritable read/write, copy-on-write, or as neither.

Mach is a distributed operating system. A copy of the Mach kernel runs at each participating node. The kernels cooperate to provide a single unified distributed system. The network message server passes Mach IPC messages between machines.

4.2.10.1.10 Project Support Environment Interfaces

No explicit features

4.2.10.1.11 Reliability, Adaptability, and Maintainability Interfaces

Reliability and Fault Tolerance - Camelot and Avalon

Camelot is a distributed transaction processing facility built on top of Mach. As such, it addresses the requirements of reliability and fault-tolerance. Its basic abstraction is the transaction. A transaction is a collection of operations that exhibits three properties: atomicity, permanence, and serializability.

Avalon is built on top of Camelot and Mach. It is implemented as a preprocessor for C++. It provides language support for reliable distributed systems based on atomic transactions.

4.2.10.1.12 Resource Management Interfaces

Storage Management

Mach places major emphasis on virtual memory management, especially in the areas of portability, advanced functionality, and memory/communication integration. In regard to portability, Mach virtual memory management assumes minimal hardware support, and is carefully constructed to isolate machine-dependent code into a single module. Not ably, it achieves improved performance, even while it minimizes hardware dependencies.

In regard to advanced functionality, Mach supports large, sparse virtual address spaces; memory mapped files; shared libraries; copy-on-write virtual copy operations; copy-on-write and read/write memory sharing between tasks, through inheritance (which is specified on a per-page basis as shared, copy, or none) of memory regions from a parent task to a child task; and user-provided memory objects and pagers.

In regard to memory/communication integration, the Mach project emphasizes the complementary roles that memory and communication can play. Namely, Mach uses memory mapping techniques (i.e., copy-on-write sharing) to accomplish communication; an entire address space may be sent in a single message with no actual data copy operations performed. In the

other direction, Mach implements virtual memory through its IPC facilities; in particular, it maps process addresses onto memory objects, which are represented by ports and accessed via messages. This is what enables user-provided memory objects.

4.2.10.1.13 Synchronization and Scheduling Interfaces

Real-Time Mach

Mach provides constructs for protection of critical regions and synchronization. Lock and unlock primitives are used with mutex variable to provide mutual exclusion. Wait and signal primitive are used with condition variables to provide synchronization.

At a higher level, Matchmaker and MIG provide a synchronous/asynchronous RPC interface.

Real-Time Mach provides an integrated time-driven scheduler, with support for both periodic and aperiodic threads. Rate monotonic scheduling policies are used for periodic threads. Value function scheduling policies (derived from Locke's thesis, as was Alpha's) are used for aperiodic threads. Real-Time Mach uses piecewise linear approximations to continuous value functions for efficiency. For a collection of periodic and aperiodic threads, the periodic threads are scheduled first, and then the aperiodic on a best effort basis.

Real-Time Mach implements policy/mechanism separation. Currently, seven scheduling policies are implemented. Different applications or experiments can utilize different policies.

Tools and a test bed have been developed to support Real-Time Mach. They allow workloads to be specified, and schedules to be constructed, examined, simulated, and monitored.

Currently, Real-Time Mach has been applied only in a uniprocessor environment and only to CPU scheduling. Plans call for it to be applied in a multiprocessor environment and to other resource types (e.g., memory, I/O). Also, impacts of interactions (requiring synchronization) among threads remain to be considered.

4.2.10.1.14 System Initialization and Reinitialization Interfaces

4.2.10.1.15 Time Interfaces

Based on UNIX.

4.2.10.1.16 Ada Language Support Interfaces

Mach threads (light-weight processes) could be used to handle Ada tasking.

4.2.10.2 Additional Characteristic

4.2.10.2.1 Proprietary or Open

Mach is open. It has been widely distributed (to over 200

institutions, 2/3 of which are corporations, 1/3 universities).

4.2.10.2.2 Qualification as a Standard

In initiating the Mach project, DARPA aimed to capitalize on the de facto standard status of UNIX. Mach's UNIX compatibility is fundamental to its success and popularity. The Mach project is meant to rebuild the core of UNIX while retaining its external interfaces.

DARPA is participating in the various UNIX standardization efforts, such as POSIX and OSF. It definitely wants to exert its influence and make Mach a dominant operating system. Inasmuch as UNIX qualifies as a standard, Mach also does.

4.2.10.2.3 Platform Flexibility

As pointed out in the introduction, Mach was designed to extend UNIX functionality to multiprocessor architectures, ranging from (1) uniform access, shared memory multiprocessors (UMA, for Uniform Memory Architecture) (e.g., Encore Multimax, Sequent Balance), to (2) differential access, shared memory multiprocessors (NUMA, for non-UMA) (e.g., BBN Butterfly, IBM RP3), to (3) multicomputer architectures (NORMA, for No Remote Memory Access Architecture) (e.g., hypercube). Mach was designed to extend UNIX functionality to large memory architectures. Mach was designed to extend UNIX functionality to distributed computing environments, in which diverse architectures (i.e., uniprocessors, multiprocessors) interconnected by high speed networks support distributed applications.

Mach has achieved high portability. It typically takes less than three man-months to port Mach to a new hardware base.

Mach's performance has been measured and compared to that of other operating systems. Initial indications are that its performance is generally competitive with other UNIX implementations such as SunOS, and markedly better in some cases (fork operation, large compilation). Its multiprocessor performance has also been measured and shown to be competitive with, for example, other Sequent and Encore operating systems. A key to Mach's performance gains is its implementation of virtual memory and its integration of virtual memory and communication.

4.2.10.3 Reference:

[ACCET86], [BARON88], [COOPE87], [DRAVE88], [JENSE85a], [JONES86], [LEBLA88], [LEHOC86], [LOCKE86], [RASHI87a], [RASHI87b], [SANSO86], [SPECT88], [TCSEC85], [TEVAN87], [TIS88], [TOKUD87], [TOKUD88a], [TOKUD88b], [YEE88], [YOUNG87]

4.2.11 MTOS Survey Summary

4.2.12 RSS/M Survey Summary

The Runtime Support Software/Minicomputer (RSS/M) is a United

States Navy standard executive system for the U.S. Navy standard general purpose, 16-bit computers (the AN/UYK-44, AN/AYK-14, AN/UYK-20, and AN/UYK-20A). RSS/M is comprised of four components: SDEX/M - Kernel real time executive SCS/M - CMS-2 supervisor CROS/M - SPL/I(M) common real time operating system CIOS/M - Common Input/Output (I/O) subsystem.

RSS/M executes in a multileveled structure composed of the kernel level, the supervisor level, and the user level. The supervisor level consists of a supervisor, I/O subsystem, user written supervisor programs, and user written device drivers. The SDEX/M kernel executes programs in the context of parallel, asynchronous tasks.

SCS/M and CROS/M provide access to all supervisor and kernel level services.

CIOS/M provides management of a complete set of I/O facilities. It is device independent and does not contain drivers for specific devices. Device drivers are provided by the user. An USH-26 device driver is available for the AN/UYK-44 target.

FDEX is a non-standard executive based on SDEX/M targeted at the AN/AYK-14. This executive provides faster response through lower executive overhead by allowing access to kernel functions by user programs located at the supervisor level. It uses the same structure as SDEX/M and is not described in detail in this summary.

4.2.12.1 Operating System Service Classes

4.2.12.1.1 General Requirements

The user programs previously mentioned are programmed in CMS-2, 16-Bit Assembly Language (MACRO), or SPL/I. SCS/M supports the MTASS/M and MTASS CMS-2 and 16-Bit Assembly languages. CROS/M supports the SPL/I language.

4.2.12.1.2 Architecture Dependent Interfaces

Hardware - RSS/M is built to run on U.S. Navy standard 16-bit computers. Software - RSS/M supports a task structure. That is, segments of software which cooperate to perform a desired function. Tasks are dispatched according to priority and a set of dispatching rules.

4.2.12.1.3 Capability and Security Interfaces

RSS/M does not provide security beyond that provided by the 16-bit computer hardware. During system generation, the user determines the hardware protection for each page address register. These page address registers are loaded by SDEX/M during system initialization or reinitialization.

4.2.12.1.4 Data Interchange Interfaces

RSS/M does not provide data interchange services.

4.2.12.1.5 Event and Error Management Interfaces

Event Management

RSS/M is an event driven system. An event occurs due to one of the following: a hardware interrupt, passage of time, an SDEX/M detected error, termination of a task, or at user request. When the event occurs, the Event Management System (EMS) or specialized event facilities responds as follows: ignores the event, activates a task, signals a semaphore, or some combination of the above.

EMS identifies every event with a unique event ID, provides event registration or cancellation, and performs the requested actions when the event occurs. Event tasks have higher priority than non-event tasks and interrupts at its class or below are locked out. A supervisor program or user task registers with EMS for an event. This registration defines: its class (priority), event type (normal or timed), whether a timed event is time-critical or time-dependent, when a timed event is to occur and the period between occurrences, event processing requested (e.g., signal semaphore), address of task to be activated, and parameter to be passed to the event task.

When an event occurs that is registered with EMS, the action requested during registration is performed. This includes signalling a semaphore, activating an event task, cancelling a non-recurrent event, and/or scheduling the next occurrence of a recurrent event.

Error Management

Error management of hardware errors (i.e., memory resume, parity, or protection errors) and software errors are handled by two functions: the interrupt handler and the error handler.

The interrupt handler processes interrupts using two methods. One method generates an event in response to a hardware interrupt and the other method allows the user to specify an interrupt handler which captures the interrupt. The event generated by an interrupt runs in executive mode with all Class 1 and 3 interrupts of equal or higher priority class locked out. The user interrupt routine is used in situations where timely interrupt response is required. This routine captures an interrupt and receives control with all interrupts locked out. If an interrupt does not have an event or captured handler associated with it, SDEX/M provides default event processing which either performs as cold start of the system or returns control to the event dispatcher.

The error handler receives control when an Executive Service Request (ESR) provides incorrect parameters. The error causes a Class 1 or 2 event to be scheduled depending on the state of the executive at the time of error detection. Different event subclasses differentiate between fatal, warning, and cautionary errors. Application systems register for all possible error events indicating the event task to be scheduled. If there is no registration for an event, SDEX/M provides default event processing which either performs a cold start of the system or returns control

to the event dispatcher.

4.2.12.1.6 File Interfaces

RSS/M provides no file services.

4.2.12.1.7 Generalized I/O Interfaces

SCS/M provides access to the I/O procedures of CIOS/M and CIOS/M compatible device drives. This is an optional feature which manages a set of I/O operations as well as providing direct access to device drivers. Device drivers, except for the AN/USH-26 cartridge magnetic tape, are written by users.

CIOS/M provides a task with a set of general I/O operations and manages I/O requests. The services provided allow the user to open a device, close a device, restart a device, and queue an I/O request.

CIOS/M provides and manages a set of device independent I/O operations. An application system can include device handlers to communicate with an operator terminal through CIOS/M operations.

4.2.12.1.8 Networks and Communications

No explicit features

4.2.12.1.9 Process Management Interfaces

Task management supports the segmentation of software into small, independent units which cooperate to perform a desired function. Task management controls scheduling, dispatching, and task state. Tasks are dispatched according to priority and a set of dispatching rules. The execution time environment for a task is established and maintained by this function.

Each task has a number of characteristics. These characteristics are: a. Tasks exist in one of two major states: blocked and unblocked. b. Each task is assigned a priority. The running task is always the highest priority unblocked task. c. Tasks can have either normal dispatching or round-robin dispatching. Round-robin tasks share the CPU with other round-robin tasks, each executing for a period of time. d. Tasks have assigned dispatch order. For tasks of the same priority, the dispatch order is determined by the order in which they were created. e. In multiprocessor configurations, tasks have a processor affinity. That is, the task runs on the master only, the slave only, or by any available processor ("don't care"). f. Tasks are either user activated or event activated.

Since all unblocked tasks must be assumed to be running all the time, semaphores and critical sections are used to control accessed to shared data and resources.

4.2.12.1.10 Project Support Environment Interfaces

RSS/M systems are built and supported by standard U.S. Navy support software: MTASS/M, Level 2, and MTASS. All three systems generate the SDEX/M executive tables used to specify system configuration.

4.2.12.1.11 Reliability, Adaptability, and Maintainability Interfaces

Reliability

RSS/M provides an error management function to detect and recover from errors.

Adaptability

Some SDEX/M capabilities are selectable by the user at SDEX/M compile time by the use of compile time parameters. In addition to the processor configuration and data structure sizing, the compile time parameters select memory management support, cautionary error checking, history recorder, and the In-Flight Performance Monitor (IFPM). Additionally, users may write their own supervisory programs and I/O device drivers to extend RSS/M capabilities.

Maintainability

The history recorder function of SDEX/M and SCS/M is provided as a debug tool for the diagnosis of system failures and also as a debug tool to aid software development. The history recorder generates a runtime log of system usage which may be analyzed off-line by the Performance Monitor tool to gauge system performance. The log contains records for task dispatch, interrupts, events, ESRs, and other supervisor defined activities. Each record identifies the record type and information specific to the record (e.g., dispatch address, task ID, external interrupt data, semaphore value). The Performance Monitor provides a summary of CPU usage for all tasks and information regarding I/O channel utilization.

The In-Flight Performance Monitor (IFPM) provides AN/AYK-14 users to ability to verify the functionality of hardware modules.

4.2.12.1.12 Resource Management Interfaces

Memory Management

Memory management in the SDEX/M versions targeted for expanded memory 16-bit computers supports use of both physical memory in excess of 64K and memory access protection. Memory access tables provide virtual address to physical address translation. Additional services allow the user to modify this translation.

This function controls the task address space and the supervisor address space. The task address space is prepared when a task is dispatched, when a supervisor program requests a change, and when a cold start initialization is performed. SDEX/M also supplies Kernel Service Requests (KSRs) to modify the virtual memory requirements defined at system generation time. Supervisor

programs can access an arbitrary area of memory through a window. Transient segments are provided to allow virtual overlays.

Memory management in AN/AYK-14 dual processors divides the available task address space using the task processor affinity characteristic.

Device Management

SCS/M provides access to the I/O procedures of CIOS/M and user written device drivers. CIOS/M provides and manages a set of device independent I/O services available to the supervisor program and the applications which it supports. Physical device drivers use CIOS/M services: open a device, close a device, restart a device, and queue an I/O request. User I/O requests select read, write, and control operations, and specify completion processing. Completion processing can include signalling a semaphore or having user specific processing performed.

4.2.12.1.13 Synchronization and Scheduling Interfaces

Synchronization Services

RSS/M provides for the user with software semaphores. Semaphores are used to block and unblock tasks. The standard completion procedure for CIOS/M (STDCMSCP) signals a semaphore upon completion of an I/O request.

RSS/M also provides critical section support. A critical section raises the current task's priority so it can only be preempted by a Class 1 event task.

Scheduling

RSS/M scheduling services are discussed in the Process Management section above.

4.2.12.1.14 System Initialization and Reinitialization Interfaces

RSS/M is initially started by bootstrap loading into the 16 bit computer. A user specifies an initial task to be activated or an initial event to be registered. During both initialization and reinitialization, RSS/M clears all tasks, events, and semaphores from its executive tables, and then starts the initial task or event. A user can specify if reinitialization should execute a cold start (user variables are not reinitialized) or a bootstrap load.

4.2.12.1.15 Time Interfaces

A task can register timed events. These events are associated with a time value (either an absolute time or a length of time in the future) at which the event is caused. Periodic events can also be registered.

Supervisor level software can also register for time-outs. A time-out functions like a non-persistent time critical event except it is associated with a supervisor procedure rather than an event task.

RSS/M uses the 16-bit computer's real time clock (RTC) and monitor clock to provide these entrances. ESRs are provided to read, change, and adjust the RTC value.

4.2.12.1.16 Ada Language Support Interfaces

There is no programming support for Ada.

4.2.12.2 Additional Characteristics

4.2.12.2.1 Proprietary or Open

Sponsored by NAVSEA PMS 412.

4.2.12.2.2 Qualification as a Standard

RSS/M is a U.S. Navy standard for 16 bit computers.

4.2.12.2.3 Platform Flexibility

RSS/M is targeted at only U.S. Navy standard 16 bit computers.

4.2.12.2.4 Application Domain

There are currently 16 different projects using RSS/M. These projects include DDG-51, PLRS, EPLRS, and ACDS.

4.2.12.2.5 Testability and Performance Evaluation Mechanism

The history recorder function of SDEX/M and SCS/M is provided as a debug tool for the diagnosis of system failures and also as a debug tool to aid software development. The history recorder generates a runtime log of system usage which may be analyzed off-line by the Performance Monitor tool to gauge system performance. The Performance Monitor provides a summary of CPU usage for all tasks and information regarding I/O channel utilization. The In-Flight Performance Monitor (IFPM) provides AN/AYK-14 users with the ability to verify the functionality of hardware modules.

4.2.12.3 References

[GESAL89]

4.2.13 SDEX/44 Survey Summary

The SDEX/44 real time executive system is a United States Navy standard executive system for the U.S. Navy standard general purpose, 16-bit computers (the AN/UYK-44, AN/UYK-20, and AN/UYK-20A). SDEX/44 will also support the AN/UYK-44 Enhanced Processor. SDEX/44 is part of NAVSEA PMS 412's Navy Support

Software/Standard Embedded Computer Resources (NSS/SECR). The SDEX/44 system is comprised of four components: SDEX/44 - Kernel real time executive PHM - Peripheral Handler Module DM - Debug Module EPM - Error Processing Module.

SDEX/44 is designed to provide real time control functions for 16-bit programs called user modules. SDEX/44 provides basic kernel executive functions. A complete system can be formed by the addition of site-specific system functions and user modules. There are eight types of entrances within a user module. These entrance types are: - Initialization entrance - Error entrance - Time-critical immediate entrance - I/O immediate entrance - Successor entrance - Message entrance - Time-dependent entrance - Background entrance.

DM is an extension of the kernel which provides debugging tools to the user.

PHM operates as a user module and provides a library of device handlers to support a standard set of peripheral devices. In addition to handlers in this library, a user may write a device handler using the same design as a PHM handler. PHM gives real time control of these devices and provides a standard format for all I/O requests. PHM manages queueing and dequeueing on I/O channels and notification of the user when I/O has completed.

EPM operates as a user module and enhances SDEX/44 error processing.

4.2.13.1 Operating System Service Classes

4.2.13.1.1 General Requirements

The SDEX/44 system is designed to interface with user programs consisting of one or more separate modules which are programmed in MTASS/M or MTASS CMS-2 or 16-Bit Assembly Language (MACRO).

4.2.13.1.2 Architecture Dependent Interfaces

User modules can communicate with other Navy standard computers (i.e., both 16 and 32 bit standard computers) or commercial computers using the SDEX/44 I/O management function and/or PHM.

4.2.13.1.3 Capability and Security Interfaces

SDEX/44 provides support for Navy systems that must conform to nuclear safety requirements. The SDEX/44 executive, PHM, and EPM incorporate compile-time options to include or omit the nuclear safety feature. The SDEX/44 system implements a nuclear safety feature in accordance with guidelines contained in Joint Cruise Missile Project JCMPINST 8020.1 CH-2. These requirements fall into the following categories: - Separate code and data areas - Protect application/user modules from accessing each other's code and data areas - Remove dead code - Detect any attempt to access or change

a protected area.

A nuclear safe version of SDEX/44 is currently undergoing a nuclear safety audit.

4.2.13.1.4 Data Interchange Interfaces

There are no data conversion services provided by SDEX/44.

4.2.13.1.5 Event and Error Management Interfaces

I/O management can be performed by SDEX/44 or by a user written interrupt handler. A compile-time option indicates which interrupt handler should be used. SDEX/44 I/O management is interrupt driven. When an I/O interrupt is received from the computer, the SDEX/44 kernel passes CP control to the user module registered for the I/O channel. During PHM initialization, PHM registers with SDEX/44 for control of interrupts on the I/O channels associated with its device handlers. I/O processing by SDEX/44 and PHM is described in the following paragraphs.

SDEX/44 provides several means for user modules to initiate and control I/O operations. This may be done using the initiate I/O option for time-critical registration, registering to receive CP control following an I/O interrupt, calling SDEX/44 to enable or disable externals interrupts on an I/O channel, or calling SDEX/44 to initiate an I/O transfer.

For I/O interrupt registration, a user module selects an immediate and/or a successor entrance. When an I/O interrupt is received from the 16-bit computer hardware, SDEX/44 looks at the user registration for the interrupt. If an immediate entrance is selected, the user module receives CP control at the entrance. If a successor task entrance is requested, SDEX/44 places the successor task on a scheduling list.

If a user module calls PHM with an I/O request, PHM will initiate the I/O to/from the selected peripheral device or will place it in a queue if the device is already busy. Once the I/O request completes, the user is notified by scheduling a successor and/or message task, resuming a suspended successor task, or signalling a semaphore.

Error Management

Error management includes detection of hardware and software errors in the system. The SDEX/44 error management functior allows users to register for control when an error is detected. If EPM is included in the system, it registers for all errors recognized by SDEX/44 and the user can register with EPM for control when an error is detected. EPM maintains a round-robin table of all errors that have occurred. If no user has registered for control, EPM provides default error processing. If DM and EPM are not included in the system, SDEX/44 uses the computer's Jump Keys to notify the user when an error occurs.

SDEX/44 supports three error processing options: - Return to the point of error - Return to the top of the scheduling loop -

Reinitialize the system.

The errors detected by the error management function include: a. Class I (Hardware errors): power tolerance and memory resume. b. Class II: instruction fault, floating point over/underflow, real time clock overflow, executive mode fault, and memory protect fault. c. Class III: I/O interrupt on unregistered channel and user I/O queue overflow. d. Required/Optional Executive Service Request (ESR) errors: errors generated by an ESR call.

4.2.13.1.6 File Interfaces

PHM provides the capability for a user module to generate a file handler. The user would act as a intermediary for all accesses to the file device and would call PHM to perform the I/O requests to the physical device.

4.2.13.1.7 Generalized I/O Interfaces

As stated in event services, SDEX/44 provides the means for user modules to initiate and control I/O operations. This includes initiating I/O in response to a hardware interrupt, registering to receive CP control following an I/O interrupt, enabling or disabling external interrupts on an I/O channel, or calling SDEX/44 to initiate an I/O transfer.

During PHM initialization, PHM registers with SDEX/44 for control of interrupts on the I/O channels associated with its device handlers. PHM gives real time control of its peripheral devices and provides a standard format for all I/O requests. PHM manages queueing and dequeueing on its I/O channels and notification of the user when I/O has completed. Queued I/O requests are processed on a first-in, first-out (FIFO) basis. PHM receives control from SDEX/44 when the I/O transfer completes. The user is notified by scheduling a successor and/or message task, resuming a suspended successor task, or signalling a semaphore.

Man-Machine Interface (MMI) Services

PHM provides keyboard devices which allow an operator to interface with user software. PHM provides the capability to modify channel assignments for the physical devices it controls. PHM also allows an operator to send a message to a user module. Operator response is also requested when an error is detected on a peripheral device (e.g., when a magnetic tape unit is off line).

4.2.13.1.8 Networks and Communications

There are no network capabilities within SDEX/44.

4.2.13.1.9 Process Management Interfaces

SDEX/44 uses a layered task dispatching algorithm in which the different task types (i.e., successor, message, time-dependent, and background) are checked for dispatching in a user specified order. a. Successor tasks receive control in response to a user

request. b. Message tasks receive control to process messages from other tasks. c. Time dependent tasks receive control on a time related basis. d. Background tasks receive control on a time available basis.

Selection of a task for execution is based upon its tier priority such that all tasks of a specified type are given CP processing time after all tasks of a higher priority type are completed and before tasks of a lower priority type are begun. When no tasks are located for execution in the lowest priority task type, the scheduling function resumes its search at the highest priority task type. Designation of task type priorities is accomplished by using SDEX/44 compile-time parameters defined by user requirements. By using a compile-time parameter, users may select or omit each of the possible task types. Also, the number of each task type is selectable at compile-time.

Successor tasks are processed using the priorities designated when the tasks were scheduled with SDEX/44. Tasks designated as having the same priority are processed on a FIFO basis at their particular priority level. There is a maximum of 31 separate priority levels, but a compile-time parameter defines the exact number of levels. Successor tasks are placed on the scheduling list in three ways: in response to an I/O interrupt, in response to a time-critical interrupt, or by being scheduled by a task. A successor task can be preempted if a higher priority successor is scheduled in response to an I/O or time-critical interrupt, or by an executing task. Binary and counting semaphores can also be used to synchronize successor task operation. There are ESRs which allow a successor task to wait on a semaphore and allow a task to signal a semaphore. PHM has a completion option which signals a successor task when an I/O operation completes. SDEX/44 also supports reentrant successor tasks. Each successor receives a unique data area which is not initialized before the task receives control from SDEX/44.

Message tasks are processed on a FIFO basis. There are system and local messages in SDEX/44. System messages provide the capability to pass data which was previously stored in an SDEX/44 storage area. Local messages provide the capability of passing data which was previously stored in a storage area which is shared by the sending and receiving modules.

A time-dependent task is selected to receive control strictly on a round-robin basis. Timing parameters associated with time-dependent tasks are used by the scheduling function to determine when the tasks are due for execution. These tasks can be recurrent at user specified rates or non-recurrent.

Background tasks are processed on a round-robin basis and receive control on a time-available basis. These tasks are selected for execution in the same manner as time-dependent tasks with an additional parameter which defines the time between suspensions. However, background tasks are unconditionally suspended when a monitor clock or I/O interrupt is received. These tasks can be recurrent at user specified rates or non-recurrent.

DM provides an operator interface to keyboard devices

allowing the operator to inspect and change memory locations, examine executive tables, enter software breakpoints or snapshots, or write a bootable system tape.

LPOS to LPOS Communication Services

Through the SDEX/44 I/O management function and PHM, users can communicate with other Navy standard or commercial computers. The format of the data is defined by the user.

4.2.13.1.10 Project Support Environment Interfaces

SDEX/44 systems are built and supported by standard U.S. Navy support software: MTASS/M, Level 2, and MTASS. All three systems generate the SDEX/44 executive tables used to specify system configuration.

4.2.13.1.11 Reliability, Adaptability, and Maintainability Interfaces

Reliability

The SDEX/44 system supports a nuclear safety feature, and also provides an error management function to detect and recover from errors.

Adaptability

Many of the capabilities and features of the SDEX/44 system are selectable by the user at SDEX/44 compile-time by the use of compile-time parameters which define a unique version. These parameters allow the user to select only those features within the SDEX/44 kernel, DM, PHM, and EPM which support specific configuration needs. Life cycle support of the SDEX/44 system includes five versions of the executive. By excluding unneeded capabilities, the memory requirements of the SDEX/44 system can be decreased.

There are currently 123 different versions of SDEX/44 which have been delivered to users. Currently, there are 44 projects actively using SDEX/44. These projects include Enhanced Modular Signal Processor (EMSP), Aegis cruiser, and Vertical Launch System (VLS).

Maintainability

SDEX/44 provides resource monitoring tables which collect data on task entrances, interrupt history, and Executive Service Requests. These tables, plus the EPM error log, provide a history of system operation. Additionally, system operation can be monitored by using the Module History feature which collects data pertaining to all tasks associated with a particular user module or the Successor History option which provides the history for an individual successor task.

During SDEX/44 initialization, a subset of AN/UYK-44 built in tests (BIT) can be run to verify the hardware.

As a Navy standard, SDEX/44 users receive life cycle support

for their project. This support includes new revision releases, telephone support, Support Software Trouble Report and Engineering Change Proposal support, and up to five new user versions.

4.2.13.1.12 Resource Management Interfaces

Memory Management

The memory management function in expanded memory SDEX/44 versions (AN/UYK-20A or AN/UYK-44) controls the virtual memory facilities within the computer. This function prepares the virtual memory environment before a user module receives CP control and supplies ESRs to modify the virtual memory requirements defined at system generation time (i.e., page register protection provided by the hardware, transient segments). Transient segments are provided to allow the virtual overlays.

Additional ESRs support dynamic runtime memory allocation. SDEX/44 manages a pool of user defined free physical memory. From this pool, users may allocate blocks of physical memory and then define the blocks as a new part of a user module or as belonging to a successor task. Through this method, users may redefine and expand/decrease their modules at run time. Successor task reentrancy, which provides unique data segments for each entrance of a successor task, is also supported by the bank of free physical memory.

Device Management

PHM provides a standard format for all I/O requests regardless of device type. Secondary memory management on disks and magnetic tapes is available using the PHM device handlers. These handlers allow users to access data on these devices.

4.2.13.1.13 Synchronization and Scheduling Interfaces

The scheduling services provided by SDEX/44 are discussed in the Process Management section above.

Binary and counting semaphores can be used to synchronize successor task operation. There are ESRs which allow a successor task to wait on a semaphore and allow a task to signal a semaphore. PHM has a completion option which signals a successor task when an I/O operation completes.

4.2.13.1.14 System Initialization and Reinitialization Interfaces

The SDEX/44 system initialization provides the means to set SDEX/44 and user modules to their initial state during computer start-up or restart. For expanded memory systems such as the AN/UYK-20A and the AN/UYK-44 which have multiple page address register (PAR) sets, the page registers are loaded with the values specified by the user at system generation time (i.e., MTASS/M SYSGEN, or MTASS Linkage Editor and Tape Builder/M).

Once the SDEX/44 kernel completes its initialization, CP control is given to each user module to initialize its state. A

module may schedule task entrances and other responsibilities by requesting executive services. In addition, modules may set themselves to initial processing states. Once the module has initialized itself, it returns CP control to SDEX/44. When all modules are initialized, the SDEX/44 initialization function passes CP control to the SDEX/44 scheduling function to begin system processing.

4.2.13.1.15 Time Services Interfaces

SDEX/44 uses the clock on a 16-bit computer (which measures time in milliseconds) for time related entrances (time-dependent), time available entrances (background), as well as time-critical entrances. The time-dependent, background, and time-critical entrances can be recurrent at user specified rates or non-recurrent.

For time-critical entrances, the computer's monitor clock is used to generate a hardware interrupt, the time-critical immediate entrance receives CP control at the user specified time. In addition to the immediate entrance, a time-critical registration also allows a successor task to be scheduled and/or I/O to be initiated.

In addition, a successor task can suspend its operation for a fixed time interval. Before the task suspends itself, it must schedule a time-critical or time-dependent entrance. The time to begin execution at the entrance specifies the suspension time interval. When the entrance receives CP control, it resumes the suspended successor task by calling SDEX/44.

ESRs are also provided to allow the user to update the time values associated with time dependent, background, and time-critical entrances, or to update the computer's clock value.

4.2.13.1.16 Ada Language Support Interfaces

There is no programming support for Ada.

4.2.13.2 Additional Characteristics

4.2.13.2.1 Proprietary or Open

Sponsored by NAVSEA PMS 412.

4.2.13.2.2 Qualification as a Standard

SDEX/44 is a U.S. Navy Standard Executive for 16-bit computers.

4.2.13.2.3 Platform Flexibility

SDEX/44 is targeted at only U.S. Navy Standard 16-bit computers.

Application Domain

There are currently 123 different versions of SDEX/44 which have been delivered to users. Currently, there are 44 projects actively using SDEX/44. These projects include Enhanced Modular Signal Processor (EMSP). Aegis cruiser, and Vertical Launch System (VLS).

Testability and Performance Evaluation Mechanism

To aid a user during project development, DM and EPM provide enhanced debugging support. EPM enhances the SDEX/44 error management function by maintaining a round-robin table of all errors that have occurred during system execution and providing default error processing. DM receives CP control when an unregistered error is detected by SDEX/44 or an operator has entered a software breakpoint. DM allows the user to examine the state of the system by providing dumps of executive tables, history tables, the EPM error log, memory areas, and hardware status. In addition, DM provides software breakpoints, snapshots, and patching capability. This information can be displayed on either a display or a printer by using PHM or DM device handlers. In addition, the contents of memory locations can be displayed or modified by using DM capabilities.

Users can improve SDEX/44 system performance by removing unneeded features, such as debugging aids. These features are controlled by compile-time options within SDEX/44, PHM, DM, and EPM.

4.2.13.3 References

[HALEE89]

4.2.14 SDX Survey Summary

The Standard Distributed Executive (SDX) is a real time executive which provides the capability to operate in a multicomputer environment via the Shipboard Integrated Processing and Display System (SHINPADS - Note: SHINPADS is a registered trademark of the Canadian Department of National Defence) Serial Data Bus (SDB) network. The SDB network supports up to 255 computers connected via a 10 MBPS electrical bus. SDX supports the standard U.S. and Canadian Navy 16 bit general purpose computers (AN/UYK-44, AN/UYK-20, AN/UYK-20A, AN/UYK-502, and AN/UYK-505). SDX is also SDEX/44 compatible. Some SDEX/44 programs can run under SDX. SDX programs are generally not able to run under SDEX/44. The SDX based Distributed Operating System (DOS) is comprised of four components:

SDX - Kernel real time executive
DSMR - Dynamic System Monitoring and Reconfiguration Module
SL - System Loader Module
BOOT - Bootstrap loaders.

SDX provides real time and time critical control functions for software units called modules. Systems or programs are

comprised of modules. Each module may have multiple types of entrances for CP control to be received from the executive. SDX supports the following types of module entrances: time critical, message, Input/Output (I/O), successor, time dependent, and background.

SDX provides the following types of functional capabilities:
Initialization
Scheduling
I/O management
Interrupt management
Error management
Bus Handler
Resident Monitor
Resident Loader
Executive Service Requests.

SHINPADS SDB systems implement hardware and software redundancy and reconfigurability to provide fault tolerance. DSMR provides real time error detection, error isolation, hardware reconfiguration, and software reconfiguration capabilities. DSMR provides a user interface to allow applications to monitor or direct reconfiguration.

SL is a loader module which supports the loading of configurations (initial BOOT load) or individual modules into remote processors via the SDB.

Bootstraps are routines which reside in Non Destructive Read Only (NDRO) memory and support the initial loading of computers via the SDB.

4.2.14.1 Operating System Service Classes

4.2.14.1.1 General Requirements

The SDX system is designed to interface with user programs consisting of one or more separate modules which are programmed in MTASS/M or MTASS CMS-2 or 16-Bit Assembly Language (MACRO).

4.2.14.1.2 Architecture Dependent Interfaces

The DOS is specifically designed to operate in a SDB network of Navy standard 16 bit computers. SDX can operate as a local executive only. SDX user modules can communicate with other Navy standard computers (i.e., both 16 and 23 bit standard computers) or commercial computers using the SDX I/O management standard 16 and 32 bit peripheral or intercomputer communication.

4.2.14.1.3 Capability and Security Interfaces

SDX supports the read, write, and execution protect functions provided by the Navy standard 16 bit computers and MTASS.

4.2.14.1.4 Data Interchange Interfaces

There are no data conversion services provided by SDX.

4.2.14.1.5 Event and Error Management Interfaces

Event Management

SDX allows user modules to register entrances to process specific interrupts and interrupt types (External Interrupt, Input Monitor, Output Monitor). When an interrupt is received, CP control is passed to the appropriate module entrance. I/O entrances may be registered as successor or time critical entrances.

SDX supports I/O related Executive Service Requests (ESRs) which allow the user module to enable/disable interrupts, initiate I/O, or change interrupt save areas for a particular channel.

Error Management

Error management includes detection of hardware and software errors in the system. The SDX error management function allows users to register for control when an error is detected. Users may request responsibility for any subset of possible errors. If no user is registered for responsibility of the type of error detected, SDX uses the computer's Jump Keys to notify the user when an error occurs. Error information is available in the computer's General Registers at the time of the error stop.

SDX supports three error processing options: return to the point of error, return to the top of the scheduling loop, or reinitialize the system.

The errors detected by the error management function include:

a. Class I (Hardware errors): power tolerance and memory resume.

b. Class II: instruction fault, floating point over/underflow, real time clock overflow, executive mode fault, and memory protect fault.

c. Class III: I/O interrupt on unregistered channel and user I/O queue overflow.

d. Required/Optional ESR errors: errors generated by an ESR call.

4.2.14.1.6 File Interfaces

The DOS does not provide file management services. The SL module does require an interfacing File Management module to support loading of modules from a mass storage device.

4.2.14.1.7 Generalized I/O Interfaces

As stated in event services, SDX provides the means for user modules to initiate and control I/O operations. This includes initiating I/O in response to a hardware interrupt, registering to receive CP control following an I/O interrupt, enabling or

disabling external interrupts on an I/O channel, or calling SDX to initiate an I/O transfer.

The DOS does not provide a man-machine interface. An application communicates operator actions to the DOS via ESR calls and implementing a user interface module which communicates with the DMSR module.

4.2.14.1.8 Networks and Communications

The DOS operates in a SHINPADS SDB system. This system supports up to 255 distributed processors connected via a linear bus. Each processor communicates to the network via a local SHINPADS node. The SHINPADS nodes are microprocessors which are responsible for bus data exchange and command protocol.

Communication between processors is via messages. Users schedule messages via ESR calls. Users are also able to register to receive selected broadcast message types. Messages are scheduled as point-to-point (selected by module number), broadcast (only registered modules will receive), or by System Identifier (each module has a unique identifier). Physical destination mapping and SDB formatting is done by the Bus Handler function of SDX.

The SDB network provides hardware redundancy of cables, nodes, and processors. The distributed network supports software redundancy as well. The DSMR module is responsible for managing these redundancies. The DSMR module provides real time, time critical system monitoring and reconfiguration capabilities which fully utilize the fault tolerance provided in the redundant distributed network.

4.2.14.1.9 Process Management Interfaces

The scheduling function of SDX is responsible for allocating the resources of the CP according to established hierarchical algorithms. Tasks are placed on various queues in response to ESRs, interrupts, or time related eligibility. Once queued, CP control is relinquished to the appropriate task by individual task dispatch processors. The scheduling function scans the queued tasks for the highest priority task according to the following priority tiers:

a. time critical
b. priority level 0 message
c. I/O
d. priority level 1 message
e. successor
f. priority level 2 message
g. time dependent
h. priority level 3 message
i. background.

In addition to the overall task priority scheme, the following

rules govern task dispatch:

j. If no task is queued and/or eligible for dispatch, control returns to the top of the dispatch loop and the process is repeated.
k. Any immediate task type may not suspend processing of any other immediate task type.
l. Any immediate task type may suspend processing of any non-immediate task type.
m. Any suspended non-immediate task type will be resumed (from point of suspension) prior to dispatching control of any other non-immediate task.
n. User suspended successor or background tasks remain suspended and ineligible for dispatch until a Resume Suspended Task ESR is performed. At that time, the task will be eligible for dispatch.

Time critical, priority 0 message, and I/O tasks are considered immediate entrances. The other task entrances are non-immediate. Priority level 0 messages generate a class III interrupt and are therefore processed as an immediate entrance.

SDX supports multiple priorities of successor tasks. The number of priorities supported is an assembly time parameter. Tasks are first-in first-out (FIFO) within tier and priority.

Time dependent tasks become eligible for dispatch based on a Real Time Clock (RTC) value. When the associated time value has expired, they become eligible based on tier priority. Time dependent tasks may be scheduled once or on a periodic basis.

Background tasks are dispatched on a time available basis. Background tasks may be scheduled for execution once or on a periodic basis. Users may specify minimum intervals between background task initiations and suspensions.

Through the SDX I/O management function, users can communicate with other Navy standard or commercial computers. The format of the data is defined by the user.

SDX is designed to operate in a distributed network with multiple copies of SDX as an LPOS communicating with each other. This is discussed in more detail in Networks and Communication.

4.2.14.1.10 Project Support Environment Services

SDX systems are built and supported by standard U.S. Navy support software: MTASS/M, Level 2, and MTASS. All three systems generate the SDX executive tables used to specify system configuration.

4.2.14.1.11 Reliability, Adaptability, and Maintainability Interfaces

Reliability

The SDX system provides an error management function to detect and recover from errors.

Adaptability

SDX and DSMR have system supplied data elements which allow the user to tailor the functionality and size to meet their specific needs. These items are all selectable at assembly time. This allows tailoring within each individual processor as well as the whole SDB network. This ability allows the DOS to support a variety of hardware and software configurations.

The SDX based DOS currently supports the Marine Air Traffic Control and Landing System (MATCALS), Canadian Patrol Frigate (CPF), Tribal Class Update and Modernization Program (TRUMP), and Infrared Search and Target Designation (IRSTD) programs.

Maintainability

SDX maintains a detailed error log pertaining to hardware and software errors detected by each processor. System operation can be monitored by using the Module History feature which collects data pertaining to all tasks associated with a particular user module or the Successor History option which provides the history for an individual successor task.

The DOS provides various health monitoring, test and loopback features to monitor and report the health of nodes, cables, and processors throughout the SDB network.

Site specific changes may be isolated to modifying system supplied data elements. This reduces risk and complexity of maintenance.

4.2.14.1.12 Resource Management Services

Memory Management

The SDX memory management function controls the virtual memory facilities within the computer. This function prepares the virtual memory environment before a user module receives CP control and supplies ESRs to modify the virtual memory requirements defined at system generation time (i.e., page register protection provided by the hardware, transient segments). Transient segments are provided to allow the virtual overlays.

SDX maintains an area of global memory. Users may assign blocks of this memory. This area may also be used for message exchange, I/O buffers or chains. Use of this area for shared memory purposes decreases the total size of program data required and can simplify the virtual memory mapping requirements. The size of the area is an assembly time parameter which may be expanded at link time.

Device Management

SDX does not directly manage any devices other than the SHINPADS node which is the network interface.

4.2.14.1.13 Synchronization and Scheduling Interfaces

The scheduling services provided by SDX are discussed in the Process Management section above.

SDX provides a synchronized system time. This is discussed in the Time Services paragraph.

4.2.14.1.14 System Initialization and Reinitialization Interfaces

The SDX system provides hardware, node, system, and local module initialization. SDX determines the type of processor it is executing in and performs the appropriate interrupt and page register initialization. SDX initializes communication with its local node and establishes SDB network communication.

Once SDX completes system initialization, SDX queues an initialization message to each user module to initialize its state. A module may schedule task entrances and other responsibilities by requesting executive services. In addition, modules may set themselves to initial processing states. Once the module has completed initialization processing, it returns control to the SDX dispatch loop. Page registers are loaded to the appropriate values as part of the dispatch function for any module task.

4.2.14.1.15 Time Interfaces

SDX uses the local hardware clock on a 16-bit computer for time related entrances (time dependent), time available entrances (background), as well as time critical entrances. The time dependent, background, and time critical entrances can be periodic at user specified intervals or can be scheduled for a single occurrence.

SDX maintains a synchronized system time which is available to users via ESR call. This provides a Greenwich Mean Time (GMT) synchronized throughout the distributed network whose increment is in milliseconds. The SHINPADS nodes support a high priority interrupt called a Time Synchronization External Interrupt. This interrupt has known propagation and transfer delays, and is therefore the mechanism used to maintain the synchronized time.

For time critical entrances, the computer's monitor clock is used to generate a hardware interrupt, the time critical immediate entrance receives CP control at the user specified time. In addition to the immediate entrance, a time critical registration also allows a successor task to be scheduled and/or I/O to be initiated.

In addition, a successor task can suspend its operation for a fixed time interval. Before the task suspends itself, it must schedule a time critical or time dependent entrance. The time to begin execution at the entrance specifies the suspension time interval. When the entrance receives CP control, it resumes the suspended successor task by calling an SDX ESR.

ESRs are also provided to allow the user to update the time values associated with time dependent, background, and time critical entrances, or to update the computer's clock value.

4.2.14.1.16 Ada Language Support Services

There is no programming support for Ada.

4.2.14.2 Additional Characteristics

4.2.14.2.1 Proprietary or Open

Property rights to the DOS are shared by Unisys Corporation, the Canadian Department of National Defence (DND), and the U.S. Navy.

4.2.14.2.2 Qualification as a Standard

SDX is not established as, nor adheres to any existing standard. It is compatible with SDEX/44 which is a U.S. Navy Standard Executive for 16-bit computers.

4.2.14.2.3 Platform Flexibility

SDX is targeted at only Navy Standard 16-bit computers.

4.2.14.2.4 Application Domain

SDX can operate as a local operating system. Its main domain is SHINPADS based fault tolerant systems.

4.2.14.2.5 Testability and Performance Evaluation Mechanism

SDX calculates and provides to the user processor utilization and bus I/O statistics at each local processor.

To aid a user during the project development, a set of distributed tools have been developed. These include:

Network Monitor

Network Monitor is a software program which communicates with a modified node to allow passive extraction of message and commands from the SDB. Network Monitor initiates extraction based on user defined trigger sequences of commands and/or messages and extracts user defined types of message and commands. Network Monitor supports pre or post event extraction. Network Monitor also allows users to send messages on the bus bases on a trigger event. Network Monitor provides a performance monitoring capability which will report SDB utilization and the mix of the types of messages and commands present on the SDB network.

Distributed Debug

The Distributed Debug allows an operator at a control console to perform debug functions in any computer in the SDB network (local and/or remote). Debug functions provided include:

Inspect and change (data or instructions)
Dump memory
Snapshot breakpoint
Trap messages

Send messages
Calculate execution time of a code segment
Read a module history
ESR trace
Create and insert patch files.

Distributed System Test and Evaluation Program (DSTEP) DSTEP is a test tool which allows isolation of hardware errors (hard or intermittent faults) to the lowest replaceable unit of equipment in the SDB bus transmission system (i.e., cable, connector, node card).

4.2.14.3 References

[COURT89]

4.2.15 Spring Kernel Survey Summary

The Spring Kernel is part of the Spring Project under research and development at the University of Massachusetts. This current research is directed by Dr. John A. Stankovic. Primary emphasis in distributed hard real-time systems and techniques is being developed on hardware and software testbeds at the university. Spring is implemented within the commercially available VRTX(tm) operating system. The kernel is used to interface to a higher level form of operating system such as VRTX.

The Spring Project views new and sophisticated applications as [RAMA89]:

- Large and complex
- Functioning in physically distributed environments
- Having to be maintainable and extensible due to their evolving nature and projected long lifetimes
- Consisting of many interacting time-critical components
- Resulting in severe consequences if logical and timing correctness are not met
- In order to achieve the major goals of high performance (i.e., the need to be fast) and predictability, the following areas are being explored in a synergistic fashion:
  - Scheduling algorithms for distributed time-critical systems
  - Operating system support for time-critical systems
  - Architectural support for time-critical systems
  - Tool support for building time-critical systems
  - Protocols for time constrained communication

The following concepts are used to create a flexible, yet predictable system:

- resource segmentation/partitioning
- functional partitioning
- selective preallocation
- a priori guarantee for critical tasks
- an on-line guarantee for essential tasks
- integrated cpu scheduling and resource allocation

o end-to-end scheduling

4.2.15.1 Operating System Service Classes

4.2.15.1.1 General Requirements

Emphasis is on language support, per se.

4.2.15.1.2 Architecture Dependent Interfaces

None are described in literature

4.2.15.1.3 Capability and Security Interfaces

None are described in literature or emphasized in the effort

4.2.15.1.4 Data Interchange Interfaces

No emphasis is placed on data interchange services in this effort

4.2.15.1.5 Event and Error Management Interfaces

Event management is accomplished by using a bounded SENDW and a RECVW prinitive or by providing enough information to the schedules so that it is handled by the schedules in pre-allocation of resources. Bounded waits of SENDW(T) and RECVW(T) prinitive are accounted for in the scheduling process. T is the maximum time wait (bounded wait) parameter. IPC messages are handled by the SENDW(T) and RECVW(T) as well as SEND and RECV (no wait), ALARM (immediate highest priority transmission), BROADCAST and TIMED-VIRTUAL-CIRCUIT. Lost messages are the responsibility of the application and no error management for IPC is provided.

No other error management services are emphasized

4.2.15.1.6 File Interfaces

Fixed sized file blocks are used

4.2.15.1.7 Generalized I/O Interfaces

I/O is divided into fast and slow I/O. Fast I/O is considered as periodic I/O and is handled by a separate processor. Slow I/O is handled by a dedicated front-end processor.

The I/O system is considered a separate processor entity from the Spring Kernel.

4.2.15.1.8 Networks and Communications

No specific network services are provided. A network is,

however, assumed to implement the distributed computer system. The global schedules adapts certain parameters and LPOS information to changing environmental conditions over the networked resources.

When messages are broadcast to nodes to attempt to find one which can guarantee to execute a task within hard real-time constraint, the task is not communicated. All tasks which can execute at one or more nodes are pre-stored at those nodes.

4.2.15.1.9 Process Management Interfaces

IPC messages are broadcast to each LPOS from the LPOS attempting to schedule a task, but needing to find an LPOS which can guarantee a task its real-time requirement.

4.2.15.1.10 Project Support Environment Interfaces

No explicit features

4.2.15.1.11 Reliability, Adaptability and Maintainability Interfaces

Reliability is enhanced since a task (which must execute logically correctly and within time constraint to avoid an application fault) is less likely to have a failure.

The meta level controller adapts the LPOSs to changing system environment conditions.

4.2.15.1.12 Resource Management Interfaces

All resources (CPU time, memory, file space, etc.) is divided into fixed-size blocks. All tasks which have widely varying worst-case times to average execution time are assumed to be subdivided into tasks which are more nearly equal in terms of such times. Fixed resource size is a key element in the Spring Kernel. Various primitive to obtain, create, delete, destroy, etc., objects and resources are provided.

4.2.15.1.13 Synchronization and Scheduling Services

The basic philosophy of the Spring Kernel is the scheduling of critical tasks such that they are guaranteed to have all necessary resources and to meet their hard real-time deadlines. scheduler, there are four, operate on separate processors; the applications run on application processors.

The lowest level is a local dispatcher for tasks. The local schedule locally guarantees that an incoming new task can meet its deadlines; it can also be used as a time planner for AI applications determining how to solve a problem(s). The third is a distributed scheduler which is used to find nodes other than the current local one which could guarantee the task meets its requirements if the local one cannot. Last is the Meta Level Controller which can adapt the various parameters and switches

scheduling algorithm as made necessary by changing system environmental conditions. The latter two are not part of the Spring Kernel itself.

4.2.15.1.14 System Initialization and Reinitialization Interfaces

No special emphasis is provided

4.2.15.1.15 Time Services

No special emphasis is provided

4.2.15.1.16 Ada Language Support Interfaces

No explicit features

4.2.15.2 Additional Characteristics

4.2.15.2.1 Proprietary or Open

Open since it is a government-sponsored academic research project.

4.2.15.2.2 Qualification as a Standard

This effort focuses on the technical paradigm, particularly the schedules, that is implemented. An application to Kernel interface or operating system to Kernel interface is not emphasized or optimized.

4.2.15.2.3 Platform Flexibility

An objective of the Spring Project is to provide a flexible, extendable distributed hard real-time system. It is considered flexible since the kernel can be used in various ways to satisfy differing systems. It is extendable since the underlying hardware is not constrained in resource types or system size.

4.2.15.3 References

[BIB88], [STAN88], [STAN 88-1], [RAMAM89], [RAMAM89-1], [HUAN 89]

4.2.16 SPRITE Survey Summary

SPRITE is a current research and development O.S. at the computer Science Division of the University of California, Berkeley, Director of the effort is John K. Ousterhout.

SPRITE is designed for a set of cooperating hosts that communicate over a network, the details of which are hidden from the user. Specifically, it targets networks or workstations.

SPRITE is an operating system designed to accommodate large memories, files, networks, etc., which may typify future distributed systems. Central to the SPRITE distributed operating system is a shared file system with distributed file caching. Also included in the research are transparent remote procedure calls as well as virtual memory management process migration and network transparency as viewed by the user.

4.2.16.1 Operating System Service Classes

4.2.16.1.1 General Requirements

Workstations considered in this research use LISP as a primary language.

4.2.16.1.2 Architecture Dependent Interfaces

Details and management of architectural dependent services are not visible to the user/application. The application considers itself to have all resources available locally.

4.2.16.1.3 Capability and Security Interfaces

Not specifically addressed in SPRITE

4.2.16.1.4 Data Interchange Interfaces

Not specifically addressed in SPRITE

4.2.16.1.5 Event and Error Management Interfaces

Not specifically addressed in SPRITE

4.2.16.1.6 File Interfaces

SPRITE distributed file system provides three kinds of file systems: local, remote, and pseudo-file-system. The pseudo-file-system implements foreign file systems, but at a user level outside of the SPRITE kernel. Directory and file warning is UNIX-like, and the name space is distributed. The single name space is shared by all SPRITE hosts in the system. Its distribution is hidden by the operating system. A prefix table mechanism at each node contains path names and host identification and is used to implement the distributed single name space. Whereas many operating systems set configuration dependent data at initialization, SPRITE readjusts the distributed file system (prefix) tables at any detection (system wide) of a workstation or network reconfiguration.

Concurrent and sequential write-sharing is permitted on files. Concurrent write-sharing courses file caching (described below) to be disabled and all file references to go to the server. Sequential write-sharing allows one user to open a file for a

write-sharing after another has finished. Delayed caching is compensated for by the SPRITE kernel. In delayed caching, the application node cache holds most recent (30 seconds maximum) data, even after file closing.

File caching of significantly large size is performed on each node to reduce file I/O bottlenecks. This feature capitalizes on the large physical memory available for implementation. Caching used some novel techniques to improve I/O performance and yet ensure that sequential file use by application delivers the correct copy of data. Data can be more current in a previous user's cache than in the file. Server-user and Server-previous user communications ensure flushing of "dirty" data blocks and delivery of clean data. After a 30-second delay, blocks are written from application cache to server cache; 30 seconds later, from server cache to disk. This policy is based on actual statistics of data file usage in UNIX at Berkely.

4.2.16.1.7 Generalized I/O Services

Not specifically addressed in SPRITE

4.2.16.1.8 Networks and Communications

IPC is not specifically addressed in SPRITE, but RPC (for LPOS-LPOS use only - see 10 above) is so addressed.

SPRITE is a distributed multiprocessor system implemented around a transparent shared hierarchical file system and based on a network such as a ten-megabit ETHERNET. The user/application should see the processes, resources, and data in a manner similar to time-shared UNIX.

4.2.16.1.9 Process Management Interfaces

Remote procedure calls (RPCs) are used when one kernel (LPOS equivalent), needs to call a procedure on a remote SPRITE node. Multiple RPCs from the calling node may exist and a receiving node may simultaneously process a number of RPCs to execute procedures. RPC is implemented on top of a special-purpose network protocol. Implicit message acknowledgment is used to reduce message overhead. RPC message fragmentation is allowed.

RPC is used only for LPOS-LPOS communication to activate execution of remote processes called by an application. No other use of RPC is made by SPRITE.

4.2.16.1.10 Project Support Environment Interfaces

Not specifically addressed in SPRITE

4.2.16.1.11 Reliability, Adoptability and Maintainability Interfaces

No explicit features

4.2.16.1.12 Resource Management Interfaces

Processes can be migrated to idle processors in the network. Upon return of the home users, foreign processes that migrated in can be migrated out to other hosts. Process migration occurs by "freezing" the process at its current node, transferring its state (registers, execution state, virtual memory, and file access) to the new node, "unfreezing" it at the new node.

Backing files are used in virtual memory management. Frozen processes virtual memory is stored in such backing files, and information about the files is transferred to the new node which can retrieve file blocks as necessary.

Virtual memory is managed using fixed blocks in the real memory and in the file system. Using the standard SPRITE file system to simplify process migration and to capitalize on server caches. Backing files are used for swapped out pages.

4.2.16.1.13 Synchronization and Scheduling Services

Not specifically addressed in SPRITE

4.2.16.1.14 System Initialization and Reinitialization Interfaces

Not specifically addressed in SPRITE

4.2.16.1.15 Time Interfaces

Not specifically addressed in SPRITE

4.2.16.1.16 Ada Language Support Interfaces

No explicit features

4.2.16.2 Additional Characteristics

4.2.16.2.1 Proprietary or Open

Open as it is government-sponsored research.

4.2.16.2.2 Qualification as a Standard

None. Time-shared UNIX is its application I/F model.

4.2.16.2.3 Platform Flexibility

No explicit features

4.2.16.3 References

[WOOD89]

4.2.17 V System Survey Summary

The V distributed operating system, developed under the leadership of David Cheriton at Stanford University, is designed for a cluster of workstations interconnected by a high-performance network. It has been running at Stanford University since 1982. It currently runs on SUN and MicroVAX workstations, which are interconnected by a 10-megabit Ethernet. Its goals can be elaborated as follows:

- o The V project strives for minimization of the kernel.
- o The V project strives for high performance, in particular, high-performance interprocess communication. o V's target application domains include real-time, interactive timesharing, and batch applications. Real-time requirements have always been a major consideration. Interactive timesharing has been the primary application, though. V is used to transform a cluster of workstations into a distributed system that offers users the same resource and information sharing capabilities traditionally provided by a centralized timesharing system. Recent work has investigated the possibility of supporting large distributed parallel applications.

The V distributed operating system consists of the following components:

- o V kernel: The design of the V kernel is based on two key concepts. The first is that a kernel should be "minimal." Namely, it should implement an interconnection mechanism between applications and system services, but not the system services themselves. Thus, interprocess communication (IPC) lies at the core of the V kernel. The second key concept is that the kernel must satisfy the following "integrity constraint": the kernel cannot depend upon the correctness of anything outside of itself for its own correctness. If the kernel fails, then it must be either the kernel's fault or the hardware's fault. This integrity constraint limits the minimization (of the kernel) that can be achieved. Currently, the V kernel includes the following servers in addition to the IPC facility: a communication server (which implements the management component of IPC), a time server, a process server, a memory management server, and a device server. However, the design is periodically re-examined to determine whether further reduction of the kernel is possible.
- o V system servers: These servers provide the traditional operating system services. They are implemented above the kernel, at the user process level, as multiprocess programs (based on lightweight processes). They are accessed through the V IPC mechanism. Current servers include a file server, a printer server, a display server, a pipe server, an Internet server, and a team server (which manages the execution of programs). Servers under development include a log server for optical disk storage and a time synchronization server.

4.2.17.1 Operating System Service Classes

4.2.17.1.1 General Requirements

The V distributed operating system offers programming support in the form of various run-time libraries. The libraries implement conventional programming interfaces such as Pascal I/O and C stdio. V also offers a set of system commands.

4.2.17.1.2 Architecture Dependent Interfaces

Internetwork Communication

V incorporates a system server known as the Internet server, which implements the DoD TCP/IP suite of protocols.

4.2.17.1.3 Capability and Security Interfaces

Regarding protection, each process is encapsulated in an address space, and can communicate with other processes only via IPC.VMTP, the transport protocol underlying V IPC, incorporates security mechanisms, including "entity domains" and encryption. In VMTP, direct communication can occur only on an intra-domain basis, thus ensuring the isolation between security levels required for mandatory access control. The idea is to have one domain per security level. Entities can belong to more than one domain, so trusted servers could communicate with users of different security levels. Encryption can be used as a mechanism to facilitate the secure authentication of subjects required for discretionary access control.

4.2.17.1.4 Data Interchange Interfaces

No explicit features.

4.2.17.1.5 Event and Error Management Interfaces

V incorporates an exception server outside the kernel. The kernel process server causes the exception-incurring process to send a message describing its problem to the exception server. The exception server then takes action, for example, invoking an interactive debugger.

4.2.17.1.6 File Interfaces

File Server

V implements file services outside the kernel via the file server, which implements a UNIX-like file system.

The file system utilizes a contiguous allocation scheme that results in most files being data contiguous on the disk.

Naming

V has a three-level naming system. At the highest level are

character-string names, which are used for permanent objects such as files. At the next level are object identifiers, which are used for transient objects such as open files. At the lowest level are entity identifiers, which identify transport-level endpoints (such as processes or groups of processes).

4.2.17.1.7 Generalized I/O Interfaces

The V project has developed a uniform I/O interface called the UIO interface as its system-level I/O interface (as opposed to its application-level I/O interface, which is implemented by the run-time libraries). The UIO interface is based on an abstraction known as the UIO object, which corresponds to an open file in conventional systems. The UIO interface provides some support for record I/O, locking, atomic transactions, and replication. It further supports the notion of optional and exceptional (escape-mode) functionality.

4.2.17.1.8 Networks and Communications

Interprocess Communication

V IPC is message-based. It has two distinguishing features. First, it is optimized for request-response behavior. Typically, a server runs as a dedicated process or team of processes. A client requests a service by sending a message to the server, and then waiting for the response. The request-response transaction (which is sometimes referred to as Remote Procedure Call (RPC) in the V literature) is considered fundamental in V. It directly implements the predominant fetch operation (typified by file read); namely, a client sends a request for data and receives the data in the server's corresponding response.

Second, V IPC supports multicast, both as a multi-destination delivery mechanism and as a binding (or logical addressing) mechanism. Multicast is considered fundamental to the implementation of problem-oriented shared memory, and has proved invaluable in the implementation of the V distributed operating system itself.

A transport level protocol, known as the Versatile Message Transaction Protocol (VMTP), has been developed to support V IPC. In addition to request-response and multicast transactions, VMTP also supports forwarding and streaming. In regard to streaming, it should be noted that VMTP, unlike other transport protocols, strives first for low delay, and then attempts to build high throughput capabilities (e.g., streaming) on top of the low delay foundation.

In part to support real-time applications, VMTP provides datagram message transactions, prioritized message transmission and delivery, and conditional message delivery (i.e., delivery only if the receiver is awaiting a message when the message arrives).

Pipe Server

V incorporates a pipe server that implements UNIX-like pipes.

4.2.17.1.9 Process Management Interfaces

Basic Abstractions

In the V literature, the V kernel is described as a "software backplane." Just as a hardware backplane provides slots, power, and communication, the V kernel provides address spaces, lightweight processes, and interprocess communication (in the form of message transactions). Thus, the basic abstractions of the V kernel are the following:

- o Address space: The V kernel separates the conventional process abstraction into two components. The first component is the address space, which holds programs (and open files).
- o Lightweight process: The second component of the process abstraction is the lightweight process, which is the locus of control within an executing program. Multiple lightweight processes may exist within an address space, and are referred to as a "team" of processes.
- o Message transaction: Processes communicate via message transactions. In the basic scenario, a client sends a request message to a server, and then blocks (awaiting a response message). The server receives the request message, performs the requested service, and then replies to the client with a response message.

Kernel Process Server

The kernel process server implements operations to create, destroy, query, modify, and migrate processes.

The V kernel is replicated at each participating network node. The kernels cooperate to provide the image of a single unified distributed system, in which processes execute in address spaces and communicate using V IPC. Kernel services (e.g., process management, memory management, communication management, device management) are themselves invoked via V IPC.

4.2.17.1.10 Project Support Environment Interfaces

No explicit features.

4.2.17.1.11 Reliability, Adaptability, and Maintainability Interfaces

Reliability and Fault Tolerance

V supports the notion of a process group as a set of processes identified by a "group identifier." The processes may reside at any node in the distributed system. The process group mechanism and multicast communication are used to implement distributed and replicated services. Both distribution and replication enhance reliability and fault tolerance.

4.2.17.1.12 Resource Management Interfaces

Storage Management

In V, an address space consists of ranges of addresses, called regions. The memory management system 1) binds regions to portions of open files (UIO objects), 2) manages physical memory as a cache for data from the open files, and 3) maintains the consistency of the cached data. The transfer of pages into the cache, as well as the mapping, is done on demand.

In part to support real-time applications, V enables programs to be specified as memory-resident.

Device Management

The kernel device server implements access to devices supported by the kernel, including disk, network interface, mouse, frame buffer, keyboard, serial line, and tape. The device server is device-independent code that interfaces between the process-level client and the driver modules for the individual devices. The device server implements the UIO interface.

Process-level servers (e.g., file server, Internet server, display server) implement extended abstractions using the basic interfaces provided by the kernel device server.

4.2.17.1.13 Synchronization and Scheduling Interfaces

In regard to processor scheduling, the kernel provides simple priority-based scheduling. In the uniprocessor case, the kernel allocates the processor to the highest priority process in the ready queue. In the multiprocessor case, a process is associated with a processor and its ready queue. The kernel schedules processes so that each processor is always executing the highest priority process in its own ready queue. In addition, the kernel periodically attempts to balance the load by changing the process-to-processor associations.

Above the kernel, a dedicated scheduler process implements a higher level of scheduling. The scheduler manipulates priorities to effect time-slicing among interactive and background processes. A number of high priority levels are reserved for real-time processes and are not subject to the priority manipulations of the scheduler.

4.2.17.1.14 System Initialization and Reinitialization Interfaces

No explicit features

4.2.17.1.15 Time Interfaces

One of the V kernel servers is a time server. The kernel time server maintains the current time of day, and it enables a process to read the time, set the time, and delay for a specified period of time. An operation for awaking a process that is delaying is also provided.

Time synchronization across nodes is implemented by a process

outside the kernel.

4.2.17.1.16 Ada Language Support Interfaces

No explicit features

4.2.17.2 Additional Characteristics

4.2.17.2.1 Proprietary or Open

The protocols and interfaces are open.

4.2.17.2.2 Qualification as a Standard

The V project emphasizes protocols and interfaces as a means of defining and building distributed systems. Efforts are underway to promulgate some of its protocols, most notably VMTP, through the DoD data communication protocol standards process. The naming and I/O protocols represent significant contributions to distributed system technology, and have played major roles in the development of the V distributed operating system. Other protocols of interest include ones for remote execution, migration, time synchronization, and atomic transactions.

4.2.17.2.3 Platform Flexibility

V is a distributed operating system designed for a cluster of workstations interconnected by a high-performance network. It currently runs on SUN and MicroVAX workstations, which are interconnected by a 10-megabit Ethernet.

V is being extended to run on shared memory multiprocessor machines. Targets include the DEC Firefly multiprocessor workstation and VMP, a shared memory multiprocessor machine designed and built at Stanford.

4.2.17.3 References

Primary [CHERI84], [CHERI85a], [CHERI85b], [CHERI86a], [CHERI86b], [CHERI87a], [CHERI87b], [CHERI88a], [CHERI88b], [CHERI88c], [CHERI88d], [FINLA87], [KANAK87], [TANEN85], [THEIM85]

4.3 Preliminary Related Standards Survey Summary

4.3.1 ARTEWG Survey Summary

ARTEWG is sponsored by ACM SIGAda to establish conventions, criteria and guidelines for Ada program components, improve the performance of those components, and provide a framework which can be used to evaluate Ada runtime systems. ARTEWG serves as a forum for users to interface effectively with Ada implementors, thereby encouraging development of runtime environments that meet users' needs. Since its formation in May 1985, ARTEWG has grown to include over 400 members.

The purpose of ARTEWG is to study sets of strategies, based on detailed investigation of the technology and requirements that will accelerate Ada runtime environments technology and then successfully incorporate that technology into the development of embedded real-time applications. The documents generated by these evaluations provide a frame work for executing and capturing the technical content of the Ada runtime environment. It is the hope of ARTEWG that these recommendations will be adopted as a masterplan.

ARTEWG consists of three working subgroups with additional task forces to consider Ada issues. The three principal subgroups:

- o Identify real-time dependencies in the language
- o Analyze real-time requirements
- o Define interfaces to standard real-time packages of auxiliary services and to required real-time kernel services.

Among the issues in the on-going discussion are the proposed changes to the Ada for the 1990s - Ada9X.

ARTEWG has the responsibility to maintain documents with the continuing process of identifying tasks which provide the technical content for enhancing/completing other documents. The baseline documentation set consists of:

Catalogue of Runtime Implementation Dependencies Catalogue of Interface Features and Options Survey of Mission Critical Application Requirements Framework for Describing Ada Runtime Environments Runtime Transportability Handbook.

Additionally, several other by-product documents providing guidelines for evaluating Ada runtime dependencies will be published. A draft proposal for a Statement of Work to create an Ada Runtime Dependencies Guide is in the works. The Guide, aimed at all segments of the Ada community, is designed to become an umbrella document, where many of the other ARTEWG documents will be summarized.

4.3.1.1 Catalog of Ada Runtime Implementation Dependencies

The Ada language was designed to avoid being tied to current technology, thus enabling the advances in technology to be readily accepted. For this reason, the Ada language was designed to be implementation independent. As a result, there are places in the language definition where implementors can decide on how a language feature is to be performed. For example, a discussion of the ways in which Ada compilers can be different in terms of real-time issues and still be valid Ada is continuing.

The Catalog of Interface Features and Options (CIFO) comprises a list of proposed interfaces for predefined packages that provide runtime services, such as dynamic priorities and non-preemptable sections. It is designed to improve the effectiveness of Ada applications and their supporting runtime environment implementations. The question of how to specify such services will be considered.

The idea that the runtime might give added facilities such as dynamic priorities must be studied. Certain issues are not specified in the language and this is still valid Ada. The language can be extended by the use of packages. The definition of these packages is at issue.

This catalogue is also being used as the basis for the Space Station Runtime Environment.

Among the deficiencies of the conventional Ada environment for an embedded system are:

no time scheduling
no dynamic properties
no predictable delays
no task/program kill/restart
no startup/termination/exits

All of the above items are required in an embedded system environment, plus multiple I/O connections (discretes, etc.); multiple timers; fast application interrupts; and application specific operating system features such as inter-processor communication and control. A user customizable runtime executive is part of a possible solution. In this case, validation issues are not raised since the compiled code is never touched and the validated runtime code never modified.

4.3.1.2 Challenge of Ada Runtime

A White Paper presenting an evaluation of the state of Ada runtime environment technology was released last fall. It provides a set of recommendations on how this technology can be elevated to a level suitable for embedded real-time applications, while maintaining the portability of Ada software.

4.3.1.3 Future Directions

ARTEWG has, thus far, been successful as a part-time volunteer industry group. It has been a significant factor in making the Ada community aware of the needed improvements in runtime environments. ARTEWG is continuing with its Plan of Action for detailed study, investigation and evaluation. However, it is looking to expand its evaluation with specific recommendations to DoD to ensure long-term improvements of Ada runtime environment technology.

4.3.2 ORKID Survey Summary

4.3.3 Open Systems Interconnection (OSI) Survey Summary

4.3.3.1 OSI Basic Reference Model

The OSI model is a framework for a set of standards being developed by the International Standards Organization to permit subsystems, which utilize heterogeneous hardware and software

computer components, of systems to interconnect and interwork. The set of standards being developed in accordance with this framework is frequently referred to as the ISO OSI standards. This framework is described in "ISO IS-7498, Information Processing Systems - Open Systems Interconnection - Basic Reference Model".

This set of standards includes those required to permit people and computer processes to interact among themselves, those required to manage the communication and interaction aspects of the system, and those required to permit the data exchange which supports the above. Standards in the first two classes are directly relevant to the work of the OSSWG in those instances when the system components (people and computers) are physically distributed. It should be noted that the IOS OSI set of standards does not directly address many of the concerns of an operating system within the bounds of a single computer, such as process scheduling and memory management. They do address many items which are of concern both within the bounds of a single computer and across computers such as program loading, program abort, fault tolerance, and naming issues.

To further explain this model the following extracts from IS-7498 are provided.

4.3.3.2 Purpose of Model

The purpose of this International Standard Reference Model of Open Systems Interconnection is to provide a common basis for the coordination of standards development for the purpose of systems interconnection, while allowing existing standards to be placed into perspective within the overall Reference Model.

The term Open Systems Interconnection (OSI) qualifies standards for the exchange of information among systems that are "open" to one another for this purpose by virtue of their mutual use of the applicable standards.

The fact that a system is open does not imply any particular system implementation, technology or means of interconnection, but refers to the mutual recognition and support of the applicable standards.

It is also the purpose of this International Standard to identify areas for developing or improving standards, and to provide a common reference for maintaining consistency of all related standards. It is not the intent of the International Standard either to serve as an implementation specification, or to be a basis for appraising the conformance of actual implementations, or to provide a sufficient level of detail to define precisely the services and protocols of the interconnection architecture. Rather, this International Standard provides a conceptual and functional framework which allows international teams of experts to work productively and independently on the development of standards for each layer of the Reference Model of OSI.

The Reference Model has sufficient flexibility to accommodate advances in technology and expansion in user demands. This

flexibility is also intended to allow the phased transition from existing implementations to OSI standards.

While the scope of the general architectural principles required for OSI is very broad, this International Standard is primarily concerned with systems comprising terminals, computers and associated devices and the means for transferring information between such systems. Other aspects of OSI requiring attention are described briefly.

As standards emerge to meet the OSI requirements, a small number of practical subsets should be defined by the standards developers from optional functions, to facilitate implementation and compatibility.

The Reference Model serves as a framework for the definition of services and protocols which fit within the boundaries established by the Reference Model.

4.3.3.3 Related OSI Standards

Concurrently with the preparation of this International Standard, work is in progress within ISO on the development of OSI standards in the following areas:

a) virtual terminal protocols;
b) file transfer, access and management protocols;
c) job transfer and manipulation protocols;
d) common application services and protocols;
e) Presentation layer services and protocols;
f) Session layer services and protocols;
g) Transport layer services and protocols;
h) Network layer services and protocols;
i) Data Link layer services and protocols;
j) Physical layer services and protocols;
k) OSI management protocols.

4.3.3.4 Scope and Field of Application

This International Standard describes the Reference Model of Open Systems Interconnection. It establishes a framework for coordinating the development of existing and future standards for the interconnection of systems and is provided for reference by those standards.

This International Standard does not specify services and protocols for OSI. It is neither an implementation specification for systems, nor a basis for appraising the conformance of implementations.

4.3.3.5 Open Systems Interconnection Environment

In the concept of OSI, a real system is a set of one or more computers, associated software, peripherals, terminals, human operators, physical processes, information transfer means, etc., that forms an autonomous whole capable of performing information processing and/or information transfer.

An application-process is an open system which performs the information processing for a particular application.

Application-processes can represent manual processes, computerized processes or physical processes.

Some examples of application-processes that are applicable to the open system definition are the following:

a) a person operating a banking terminal is a manual application-process;

b) a FORTRAN program executing in a computer center and accessing a remote database is a computerized application-process; the remote database management systems server is also an application-process; and

c) a process control program executing in a dedicated computer attached to some industrial equipment and linked into a plant control system is a physical application-process.

OSI is concerned with the exchange of information between open systems (and not the internal functioning of each individual real open system).

OSI is concerned only with interconnection of systems. All other aspects of systems which are not related to interconnection are outside the scope of OSI.

OSI is concerned not only with the transfer of information between systems, i.e. transmission, but also with their capability to interwork to achieve a common (distributed) task. In other words, OSI is concerned with the interconnection aspects of cooperation (see note at end of this section) between systems, which is implied by the expression "system interconnection."

The objective of OSI is to define a set of standards to enable real open system to cooperate. A system which complies with the requirements of applicable OSI standards in its cooperation with other systems is termed a real open system.

4.3.3.6 Modeling the OSI Environment

The development of OSI standards, i.e. standards for the interconnection of real open systems, is assisted by the use of abstract models. To specify the external behavior of interconnected real open systems, each real open system is replaced by a functionally equivalent abstract model of a real open system called an open system. Only the interconnection aspects of these open systems would strictly need to be described. However, to accomplish this, it is necessary to describe both the internal and external behavior of these open systems. Only the external behavior of open systems is retained as the standard of behavior of real open systems. The description of the internal behavior of open systems is provided in the Reference Model only to support the definition of the interconnection aspects. Any real system which behaves externally as an open system can be considered to be a real open system.

This abstract model is used in two steps.

First, basic elements of open systems and some key decisions concerning their organization and functioning, are developed. This

constitutes the Reference Model of Open Systems Interconnection described in this International Standard.

Then, the detailed and precise description of the functioning of the open system is developed in the framework formed by the Reference Model. This constitutes the services and protocols for OSI which are the subject of other International Standards.

It should be emphasized that the Reference Model does not, by itself, specify the detailed and precise functioning of the open system and, therefore, it does not specify the external behavior of real open systems and does not imply the structure of the implementation of a real open system.

The reader not familiar with the technique of abstract modeling is cautioned that those concepts introduced in the description of open systems constitute an abstraction despite a similar appearance to concepts commonly found in real systems. Therefore real open systems need not be implemented as described by the model.

Throughout the remainder of this International Standard, only the aspects of real systems and application-processes which lie within the OSI environment are considered.

NOTE: Cooperation among open systems involves a broad range of activities of which the following have been identified:

a) interprocess communication, which concerns the exchange of information and the synchronization of activity between OSI application-processes;

b) data representation, which concerns all aspects of the creation and maintenance of data descriptions and data transformations for reformatting data exchanged between open systems;

c) data storage, which concerns storage media, and file and database systems for managing and providing access to data stored on the media;

d) process and resource management, which concerns the means by which OSI application-processes are declared, initiated and controlled, and the means by which they acquire OSI resources;

e) integrity and security, which concern information processing constraints that have to be preserved or assured during the operation of the open systems; and

f) program support, which concerns the definition, compilation, linking, testing, storage, transfer, and access to the programs executed by OSI application-processes.

Some of these activities may imply exchange of information between the interconnected open systems and their interconnection aspects may, therefore, be of concern to OSI.

The International Standard covers the elements of OSI aspects of these activities which are essential for early development of OSI standards.

4.3.3.7 References

A sample of the ISO standards and standardization work of interest to OSSWG are the following:

[OSIa], [OSIb], [OSIc], [OSId], [OSIe], [OSIf], [OSIg], [OSIh].

4.3.4 POSIX Survey Summary

The IEEE Portable Operating System (POSIX) interface standard is based on earlier UNIX operating system interfaces. This standard defines an application program interface to an underlying set of operating system functions; it does not specify the structure, functions, or performance of the underlying operating system beyond the specific functionality visible at the application program interface level.

This standards committee, IEEE P1003, has created an initial version of the interface standard which completed balloting in August, 1988, and is therefore identified as IEEE 1003.1-1988. In addition, this committee includes a number of smaller working groups which are expected to present extensions or application-specific interfaces for this standard for official balloting before 6/90; draft versions of these extensions are currently available.

In this section, we describe the characteristics of the POSIX interface standard, including information from current draft extensions of the 1003.2 (Shells and System Utilities) working group, the 1003.4 Real-Time Extensions working group (Draft 7), and the 1003.5 Ada Bindings working group, using the framework of the NGCR-OSSWG Reference Model.

4.3.4.1 Operating System Service Classes

4.3.4.1.1 General Requirements

The POSIX standard is currently expressed in terms of the C language, but is planned for language-independent revision for future releases.

4.3.4.1.2 Architecture Dependent Interfaces

By its nature, the POSIX interface definition is architecture independent; this has been a primary requirement during its definition, and one which has been rigorously enforced by the many processor vendors participating in its definition. Beyond the POSIX standard, conforming implementations are expected to provide implementation-defined services to support specific architectures, but they are constrained to do this in ways which will not violate the POSIX standard.

4.3.4.1.3 Capability and Security Interfaces

The current standard specifies file-level permissions for read, write and execute by the owner, a specified group of users, or all users. Further security facilities are planned and are under consideration by a POSIX Security working group 1003.6. Prior

to the definition of such security extensions, conforming implementations are free to extend the standard to provide appropriate discretionary and mandatory access controls required to conform to NSCS Orange Book security requirements.

4.3.4.1.4 Data Interchange Interfaces

POSIX provides a "pipe" facility for serial communications between processes using the standard file service interfaces. For this facility, each process opens its end of the pipe, which then transmits characters between them.

Additionally, the POSIX 1003.4 Real-Time Extension provides support for message passing as a form of interprocess communication (shared memory is an alternate form, which is also supported.) It views the capability of passing messages with both high and deterministic performance as being crucial in real-time systems. It implements message passing through "message queue special files," i.e, objects named within the file system (although this definition places message queues in the file system name space, this does not imply that the function need be handled by the file management portions of the operating system; neither does it necessarily imply a FIFO queueing discipline.). Message queue special files can be opened for use by multiple sending and receiving processes.

Functions

The POSIX 1003.4 message passing facilities provide the following functions:

- Creating a message queue special file.
- Opening and closing a specified message queue special file.
- Sending a message to a specified message queue special file.
- Sending a message to a specified list of message queue special files, thus providing a multicast capability.
- Receiving a message from a specified message queue special file, asynchronously or synchronously.
- Allocating (by the sender) and freeing (by the receiver) a system-provided "message buffer" to hold the message.
- Setting the values of and getting the values of attributes of a specified message queue special file.
- Getting the status of a specified message queue special file.

4.3.4.1.5 Event and Error Management Interfaces

A "signal" mechanism is provided which allows operating system or application defined events to be sent to a process. The process may elect to handle the signal asynchronously or to ignore it, in which case the process will be terminated, along with its children. If two signals arrive before the first is handled, the

first will be lost; the signal mechanism is therefore considered unreliable.

The POSIX 1003.4 Real-Time Extension provides an additional service; an event mechanism is available providing for delivery of reliable asynchronous event notifications.

POSIX 1003.4 views asynchronous event notification as being essential in real-time systems. It strives to provide a general purpose, uniform, reliable interface that has both determinism and high performance. Its asynchronous event notification facilities consist of the following: (1) event definition data structure, (2) event trap routine function prototype definition, and (3) the functions cited below. In defining an event, a user specifies an event trap routine, an application-defined event value to be passed to the event trap routine identifying the source of the event, the event class (i.e., a grouping of related events, including a priority in case of multiple event occurrences) within which the event trap routine executes, and the event class mask to be in effect during execution of the event trap routine.

Examples of predefined asynchronous events defined in the POSIX 1003.4 Real-Time Extensions standard include asynchronous I/O completion, timer expiration, message arrival, as well as user-defined events.

Functions

The POSIX 1003.4 asynchronous event notification facilities provide the following functions:

- Changing or examining the event class mask of the invoking process.
- Waiting for asynchronous event notifications for specified event classes, in one of two modes.
- Causing a specified application-defined event to be raised for the invoking process.
- Changing the number of queue entries to be used to hold events which have been raised but not yet delivered to the invoking process.
- Achieving reliable exits from event trap routines via non-local jumps.
- Associating a specified signal with a specified event class.

4.3.4.1.6 File Interfaces

The POSIX file system consists of a hierarchically organized set of files. The hierarchy consists of a set of file directories which in turn contain pointers either to other directories or to individual files. File descriptors may be entered into more than one directory; a file is not deleted until it is removed from the last directory in which it is entered. In addition, the file system may contain "special" files which are visible in a directory, but may or may not have representations in permanent storage and do not imply that their associated functions are performed by the file management portions of the operating system. They are used for operations which cross process boundaries, such

as semaphores and shared memory.

The POSIX 1003.4 Real-Time Extensions define three extensions to the file services for use in developing real-time and database applications. These services are called:

I Real-Time Files
II Asynchronous I/O
III Synchronized I/O.

I Real-time Files

The POSIX 1003.4 Real-Time Extensions draft views the capability of performing I/O operations with both deterministic and high performance as being crucial in real-time systems. It recognizes that contiguous files are a traditional mechanism for providing deterministic high performance I/O, since most real-time systems utilize rotating magnetic disks as their file storage media, but rather than provide a specific interface to contiguous files, it provides a more general interface to "real-time files." Of course, an implementation may choose to implement real-time files using contiguous files, but it is not forced to do so. An implementation is free to take advantage of advanced or non-traditional media that can provide deterministic high performance without relying on contiguity, within a framework provided by the POSIX 1003.4 real-time file facilities.

The approach that POSIX 1003.4 takes to real-time file support is to make some critical (performance-related) attributes of the operating system's implementation of files and I/O not only visible to applications but also to some extent application controllable (i.e., at least "influenceable," through hints related to characteristics of the application, which the operating system can take into account).

Deterministic high performance means predictable (i.e, time bounded) and very short delay times.

Functions

The POSIX 1003.4 real-time file facilities provide the following functions:

- Creating a real-time file.
- Communicating to the system desirable attributes of a specified (previously created) real-time file.
- Getting the actual attributes of a specified real-time file or of real-time files of a specified file system.
- Obtaining a suitably aligned buffer of a specified size either from a specified data area or from the system.

II Asynchronous I/O Interfaces

The POSIX 1003.4 Real-Time Extensions provide an additional capability to perform file I/O asynchronously, allowing processes to perform multiple concurrent I/O operations concurrently with computations on I/O data.

Functions

The POSIX 1003.4 asynchronous I/O facilities provide the following functions:

- Asynchronously reading and writing a specified file.
- Initiating a list of I/O requests with a single system call.
- Cancelling a specified asynchronous I/O request; or, cancelling all asynchronous I/O requests to a specified file.

III Synchronized I/O Interfaces

In addition, the POSIX 1003.4 Real-Time Extensions working group recognizes that some applications require assurance of I/O completion, particularly in database applications. It views the capability of receiving such assurance as being vital in real-time systems. The POSIX 1003.4 Real-Time Extensions draft refers to I/O that is to be done with assurance of completion as "synchronized I/O."

Two types of synchronization are defined in POSIX 1003.4:

- Synchronized I/O data integrity completion.
- Synchronized I/O file integrity completion.

Functions

The POSIX 1003.4 synchronized I/O facilities provide the following functions:

- Specifying that I/O completion for a specified file is to be (re-)defined as either synchronized I/O data integrity completion or as synchronized I/O file integrity completion.
- Requesting that all outstanding I/O requests for a specified file are to be "completed" in accordance with the definition of either synchronized I/O data integrity completion or as synchronized I/O file integrity completion.

4.3.4.1.7 Generalized I/O Interfaces

Device I/O in POSIX is handled in the same way as file I/O. Device access is made through the normal file system operations, although additional primitives are available to control devices.

4.3.4.1.8 Networks and Communications

POSIX itself does not (now) address communications directly. Conforming POSIX implementations are free to provide any level of communications control desired, including transparent distributed facilities in which the existence of multiple processors is not visible to the application.

4.3.4.1.9 Process Management Interfaces

The POSIX 1003.1-1988 standard offers "heavy-weight" process concurrency. Each process has a separate address space, and shares

file descriptors and other control structures with its parent. Operating system primitives to manage processes are very easy to use (i.e., fork() and exec()).

A proposal to add "light-weight" a concurrency model (called "threads") within a POSIX process is under active consideration by the POSIX 1003.4 Real-Time Extension working group. This would provide for multiple threads of control to exist within processes, including mutual exclusion primitives (e.g., mutex). These threads would carry very little state information (i.e., stack pointers and registers) which would make them extremely "light weight" and thus allow extremely fast implementations. They would be priority scheduled in the same way as POSIX processes (see Synchronization and Scheduling Interfaces), using the same set of possible priorities.

At the POSIX application interface, no functions are provided for inter-LPOS services. Conforming implementations are expected to provide such services transparently using normal POSIX functions, or using implementation-defined extensions.

4.3.4.1.10 Project Support Environment Interfaces

The POSIX interface description does not explicitly describe project support environment services.

4.3.4.1.11 Reliability, Adaptability and Maintainability Interfaces

The POSIX interface standard includes extensive error checking for every service, defining a complete set of error returns when errors are detected, as well as an asynchronous error facility (i.e., signals and events). Actual management of error conditions, including reconfiguration, is intended to be handled by applications using these error indications with standard POSIX functions.

4.3.4.1.12 Resource Management Interfaces

The POSIX 1003.2 Shells and Utilities working group includes user-accessible memory management services (e.g., malloc()) which provide for memory allocation within the process memory space. The POSIX process model provides for the process memory spaces to be mutually disjoint; thus, the memory management services need not be included in the POSIX kernel services.

The POSIX 1003.4 Real-Time Extension, however, views the shared memory paradigm as being an important, traditional, high-performance mechanism for interprocess communication in real-time systems. It thus supports shared memory objects as "shared memory special files," i.e., objects named within the standard file system (the use of the file system name space does not imply that management and mapping of shared memory need be implemented using the file management services of the operating system). It enables shared memory special files to be mapped into

a process's virtual address space.

Functions

The POSIX 1003.4 shared memory facilities provide the following functions:

o Creating a shared memory special file. Opening and closing a specified shared memory special file.

o Mapping (and unmapping) a specified segment of a specified shared memory special file into a process's virtual address space at a specified address.

In addition, the POSIX 1003.4 Real-Time Extension working group supports the notion that a process should be able to lock its address space, or specified regions thereof, into memory. Such a capability is viewed as being crucial to deterministic high performance, which is essential in real-time systems.

Functions

The POSIX 1003.4 process memory locking facilities provide the following functions:

o Locking and unlocking specified regions of a process's address space into memory.

4.3.4.1.13 Synchronization and Scheduling Interfaces

The POSIX 1003.4 Real-Time Extension adopts the binary semaphore as the basic means of process synchronization. It notes that the binary semaphore is a "minimal" synchronization mechanism, and that other mechanisms such as counting semaphores and monitors can be implemented on top of the binary semaphore. It supports semaphores as "semaphore special files," i.e., objects named within the file system (although it must be noted that the standard does not imply that the semaphore functions need be performed by the file management portions of the operating system. In fact, the standard explicitly defines this facility in such a way as to allow implementations to avoid system calls for successful semaphore accesses.).

Functions

The POSIX 1003.4 semaphore facilities provide the following functions:

o Creating a semaphore special file.

o Opening and closing a specified semaphore special file.

o Doing a P-operation (Dijkstra, "Co-operating Sequential Processes", 1968) on a semaphore represented by a specified semaphore special file.

o Doing a V-operation on a semaphore represented by a specified semaphore special file.

The POSIX standard (IEEE 1003.1-1988) currently does not define the process scheduling to be performed. The POSIX 1003.4 Real-Time Extension, however, views preemptive, dynamic

priority-driven scheduling as being fundamental to real-time systems. It supports two variants of preemptive, dynamic-priority-driven scheduling. The variants are distinguished by the way in which processes of equal priority are scheduled. In the first variant, runnable processes of equal priority are scheduled according to a first-in-first-out (FIFO) policy.(It should be noted that if a process sets its priority to its current priority, the process is viewed as"entering" the queue; so, it becomes the last, or newest, member of the queue, regardless of its previous position.)In the second variant, runnable processes of equal priority are scheduled according to a round-robin (RR) policy, with a specified time slice.

Functions

The POSIX 1003.4 scheduling facilities provide the following functions:

- Setting the priority of and getting the priority of a specified process.
- Setting the "scheduling policy" of and getting the scheduling policy of a specified process.

4.3.4.1.14 System Initialization and Reinitialization Interfaces

The POSIX interface specification leaves system initialization and reinitialization services to be defined by the implementation.

4.3.4.1.15 Time Interfaces

The POSIX 1003.1-1988 standard provides for interrogating and reading time and date, as well as a sleep() function to delay for a set period of time. The standard defines these functions in units which are, however, unacceptably coarse for use by real-time systems.

The POSIX 1003.4 Real-Time Extension to POSIX provides additional fine resolution interfaces to system-wide timers and to per-process interval timers that make time visible to processes and enable processes to schedule timer events in a variety of useful ways. It views such interfaces as being essential to real-time systems, which are distinguished by the significance of the role that time and timing constraints play in them.

Functions

The POSIX 1003.4 timer facilities provide the following functions:

- Setting the value of, getting the value of, and getting the resolution of a specified system-wide timer.
- Creating and destroying a per-process interval timer, based upon a specified system-wide timer and a specified delivery mechanism (signals, events, or implementation-specific).

o Setting the value of, getting the value of, and getting the resolution of a specified per-process interval timer.

4.3.4.1.16 Ada Language Support Interfaces

The POSIX Ada Binding working group, IEEE P1003.5 is currently defining an interface to POSIX from the Ada language; this interface is expected to enter formal balloting by 6/90. The Ada language interface will provide access to all POSIX functions; initially, the Ada bindings working group is targeting 1003.1-1988, but plans to target the real-time extensions from 1003.4 immediately following the initial 1003.1-1988 work. Similarly, the POSIX interface must support all Ada functionality.

As with many commercial operating system interfaces, the POSIX process semantic model currently provides a poor match to Ada tasking, although the provision (in the POSIX 1003.4 Real-Time Extensions) of asynchronous I/O is expected to significantly improve the ability of Ada implementations to support Ada tasking within the process model by removing opportunities for blocking system calls issued by one Ada task to block the entire Ada program in the POSIX process. Beyond the POSIX process model, a light-weight concurrency mechanism within a POSIX process (i.e., threads) is also under active consideration by the 1003.4 Real-Time Extensions working group; if accepted, its presence is expected to significantly improve the semantic match between POSIX and Ada tasking, greatly enhancing the suitability of POSIX to handle Ada tasking (see Process Interfaces description). In addition, the timer and event mechanisms (also defined in the 1003.4 Real-Time Extensions) can be used effectively to implement the Ada delay and exception mechanisms.

4.3.4.2 Additional Characteristics

4.3.4.2.1 Proprietary or Open

As an IEEE standard, POSIX is fully open for implementation by any operating system vendor.

4.3.4.2.2 Qualification as a Standard

The POSIX standard is controlled by IEEE (P1003 committee).

4.3.4.2.3 Platform flexibility

There are no known hardware platforms unable to support a conforming POSIX implementation

4.3.4.3 References

[POSIX89]

Section 5

References

Literature References

[ACCET86] Accetta, M., Baron, R., Bolub, D., Rashid, R., Tevanian, A., Young M., "Mach: a New Kernel Foundation for UNIX Development," Technical Report, Dept. of Computer Science, Carnegie Mellon Univ., June 1986.

[ADA83] Reference Manual for the Ada Programming Language, U.S. DOD (ANSI) MIL-STD 1815a-1983, February 1983.

[AEGIS] "AEGIS Tactical Executive System (ATES/43) User's Manual," ACD 3106B, 30 June 1988, AEGIS Shipbuilding Program (PMS400), Prepared By Naval Systems, RCA Electronic Systems Department, Government Electronic Systems Division, GE Aerospace, Moorestown, NJ 08067.

[AGRAW87] Agrawala, A., and Levi, S., "On Real-Time Operating Systems," University of Maryland Computer Science Technical Report, CS-TR-1838, April 1987.

[AGRAW89] Agrawala, A Real Time Systems, McGraw-Hill Press, 1989. To be published.

[ALMES83] Almes G.T., A.P. Black and E.D. Lazowska and J.D. Noe, The Eden System: A Technical Review, University of Washington Department of Computer Science, Technical Report 83-10-05, October 1983.

[ALMES83A] Almes, G.T., Integration and Distribution in the Eden System., In IEEE International Workshop on Computer Systems Organization (New Orleans LA), pages 62-71, IEEE, March 29-31, 1983.

[ALMES83B] Almes, G.T.; Black, A.P.; Lazowska, E.D.; Noe, J.D., The Eden System: A Technical Review., Technical Report 83-10-05, University of Washington, October, 1983.

[ANDRE82] Andre, J.P.; Petit, J.C.; Derriennic-Le Corre, H., Dynamic Software Reconfiguration in a Distributed System (Galaxie), In IEEE International Conference on Communications, ICC '82:The Digital Revolutionn (Philadelphia PA), pages 5G.4.5, IEEE, June 13-17, 1982.

[ARORA86] Arora R., Rana S. and Gupta M., Distributed Termination

Detection Algorithm for Distributed Computations, Inf. Proc, Letters, Vol. 22, No. 6, pp. 311-314, May 1986.

[BAKER86] Baker, T.P. and G.M. Scallon, "An Architecture for Real-Time Software Systems," IEEE Software, pp. 50-58, May 1986.

[BALL76] Ball, J.E., Feldman, J.; Low, J.R.; Rashid, R.; Rovner, P., RIG, Rochester's Intelligent Gateway: System Overview, IEEE Transactions on Software Engineering SE-2(4):321-328, December, 1976.

[BALL82] Ball, J.E.; Barbacci, M.R.: Fahlman, S. E.; Harbison, S.P.; Hibbard, P.G.; Rashid, R.F.; Robertson, G.G.; Steele, G.L. Jr., The Spice Project, Technical Report, Computer Science Research Review, Carnegie-Mellon University, 1982.

[BANE81] Bane, R.; Stanfill, C.; Weiser, M., Operating System Strategy on ZMOB, In 1981 IEEE Computer Society Workshop on Computer Architecture for Pattern Analysis and Image Database Management (Hot Springs VA), pages 125-132, IEEE, November 11-13, 1981.

[BARON85] Baron, Robert V., Rashid, Richard F., Siegel, Ellen H., Tevanian, Avadis Jr., and Young, Michael "MACH: A Multi-Processor-Oriented Operating System and Environment," Technical Report, Carnegie-Mellon University, Department of Computer Science, 1985.

[BARON88] Baron, Robert V., MACH Kernel Interface Manual, Computer Science Department, Carnegie Mellon University, Draft Paper, February 15, 1988.

[BAYER79] Bayer, R., Graham R., and Seegmuller G (editors), Flynn, M., Gray J., Jones A., Lagally K., Opderbeck H., popek G., Randell B., Saltzer J., and Wiehle H., Operating Systems: An Advanced Course, Springer-Verlag, Berlin, Germany, 1979.

[BBN88a] BBN Laboratories Incorporated, Operator's Reference Manual, Release 1.3, September 15, 1988.

[BBN88b] BBN Laboratories Incorporated, Programmer's Reference Manual, Release 1.3, September 15, 1988.

[BBN88c] BBN Laboratories Incorporated, Tutorial Documents, Release 1.3, September 15,1988.

[BBN88d] BBN Laboratories Incorporated, User's Reference Manual, Release 1.3, September 15, 1988.

[BERET85a] Berets, J.C., R.A. Mucci and R.E. Schantz., "Cronus: A Testbed for Developing Distributed Systems." Proceedings of the IEEE Military Communications Conference, October 20-23, 1985.

Boston, MA. IEEE Communications Society CH85CH2202-0.

[BERET85b] Berets, J.C., R.A. Mucci, R.E. Schantz and K.J. Schroder., "The C2 System Internet Experiment: Interim Technical Report No. 1." BBN Report No. 6073 prepared for the Rome Air Development Center, BBN Laboratories, October 1985.

[BERET87] Berets, James C. and Richard M. Sands, "Introduction to Cronus: A Distributed Operating System," Draft Paper, BBN Laboratories Incorporated, January 1987.

[BERNA88] J.M. Gernabeu Auban, P.W. Hutto and M.Y.A. Khalidi, M. Ahamad, W.F. Appelbe, P. Dasgupta, R.J. LeBlanc and U. Ramachandran. Clouds - A Distributed Object-based Operating System: Architecture and Kernel Implementation, European UNIX Systems User Group Autumn Conference (EUUG), October 1988.

[BERNA89] J.M. Bernabeu Auban, P.W. Hutto and M.Y. A. Khalidi, M. Ahamad, W.F. Appelbe, P. Dasgupta, R.J. LeBlanc and U. Ramachandran. The Architecture of the Ra: A Kernel for Clouds, Proceedings of the 22nd Hawaii International Conference on System Sciences, January 1989 [Also available as GIT-ICS-88/250].

[BERNA] Bernabeu Auban, Jose M., et al., "The Architecture of Ra: A Kernel for Clouds," School of Information and Computer Science, Georgia Institute of Technology.

[BERNS87] Bernstein, P.A., V. Hadzilacos, and N. Goodman, "Concurrency Control and Recovery in Database Systems," Addison-Wesley Publication Co., 1987.

[BERRY86] Berry, G., Cosserat L., The ESTREL Synchronous Programming Language, Ecole Nationale Superiere des Mines de Paris, France, March 1986.

[BIB88] Bizabani, Sara R. et al., "The Integration of Deadline and Criticalness is Hard Real-Time Scheduling," COINS Technical Report 88-82, University of Massachusetts.

[BIRMA85a] Birman, Kenneth P., "Replication and Fault-Tolerance in the ISIS System," Proceedings of the Tenth ACM Symposium on Operating Systems Principles, pp. 79-86, 1985.

[BIRMA85b] K.P. Birman and others, An Overview of the ISIS Project, Distributed Processing Technical Committee Newsletter, IEEE Computer Society, Vol.7, No. 2, October 1985.

[BLACK85] Black, Andrew P., Supporting Distributed Applications: Experience with Edcen," Proceedings of the Tenth ACM Symposium on Operating Systems Principles, pp. 181-193, 1985.

[BLAIR82] Blair, G.S.; Hutchison, D.; Shepherd, W.D., MIMAS - A

Network Operating System for Strathnet, In Proceedings of the 3rd International Conference on Distributed Computing Systems (Miami/Fort Lauderdale, FL), pages 212-217, IEEE, October 18-22, 1982.

[BLAZE76] Blazewicz, J., "Scheduling Dependent Tasks with Different ARrival Times to Meet Deadlines," Proceedings of the International Workshop on Modelling and Performance Evaluation of Computer Systems, pp. 57-65, 1976.

[BOCHM83] Bochmann, G., Distributed Systems Design, Springer-Verlag, Berlin Germany, 1983.

[BOEBERT78A] Boebert, W.E.; Franta, W.R.; Jensen, E.D.; Kain, R.Y., Decentralized Executive Control in Distributed Computer Systems, In Proceedings of COMPCON 78, pages 254-258, IEEE, November, 1978.

[BOEBERT78B] Boebert, W.E.; Franta, W.R.; Jensen, E.D.; Kain, R.Y., Kernel Primitives of the HXDP Executive, In Proceedings of COMPCON 78, pages 595-600, IEEE, November, 1978.

[BOKHA81] Bokhari, Shahid H.,"A Shortest Tree Algorithm for Optimal Assignments Across Space and Time in a Distributed Processor System," IEEE Transactions SE-7, 6, pp. 583-589, November 1981.

[BOURN83] Bourne, S., The UNIX System, Addison-Wesley Publishing Co., London England, 1983.

[BROCK86] Brock, Larry D., and Lala, Jaynarayan "Advanced Information Processing System: Status Report" Proceedings of NAECON, May 1986.

[BROWN82] Brownbridge, D.R., Marshall, L.F., and Randell, B., "The Newcastle Connection or UNIXes of the World Unite!" Software Practice and Experience, Vol. 12, pp. 1147-1162, 1982.

[CARAN89] Carangelo, Antonio, Jr., "Computer Security Products Technology Overview for Navy Program Next Generation Computing Resources (NGCR)" DRAFT, the MITRE Corporation, Bedford, MA, 01730, July 14, 1989.

[CASEY87] Casey, Thomas A., Jr., Doug Weber, and Stephen T. Vinter, "The Secure Distributed Operating System Project: Final Report," Report No. 6678, BBN Laboratories Incorporated, October 1987.

[CASPI86] Caspi, P., Halbwachs N., "A Functional Model for Describing and reasoning Time Behavior of Computer Systems, " Acta Informatica, Vol 22, No. 6, pp. 595-628, March 1986.

[CHAND83] Chandy K., Misra J. and Haas L, "Distributed Deadlock Detection," ACM Transaction on Computer Systems Vol 1, No. 2, pp. 144-156, May 1983.

[CHENG88] Cheng, S.C., J.A. Stankovic, and K. Ramamritham "Scheduling Algorithms for Hard Real-time Systems" A Brief Survey," in Stankovic, J and Ramamritham, K., Hard Real-Time Systems Tutorial, IEEE Computer Society Press 1988.

[CHERI83] Cheriton, D., and W. Zwaenepoel, "The Distributed V Kernel and its Performance for Diskless Workstations," Operating Systems Review, 17(5): 128-140, October 1983.

[CHERI84] Cheriton, David R., "The V Kernel: A Software Base for Distributed Systems," IEEE Software, pp. 19-42, April 1984.

[CHERI85a] Cheriton, David R. and Paul J. Roy, "Performance of the V Storage Server: A Preliminary Report," Proceedings of the ACM Conference on Computer Science, March 1985.

[CHERI85b] Cheriton, David R. and Willy Zwaenepoel, "Distributed Process Groups in the V Kernel," ACM Transactions on Computer Systems, Vol. 3, No. 2, pp. 77-107, May 1985.

[CHERI86a] Cheriton, David R., "Problem-oriented Shared Memory: A Decentralized Approach to Distributed System Design," Proceedings of The 6th International Conference on Distributed Computing Systems, pp. 190-197, May 1986.

[CHERI86b] Cheriton, David R., "VMTP: A Transport Protocol for the Next Generation of Communication Systems," *Proceedings of SIGCOMM* 86, pp. 406-415, August 1986.

[CHERI86c] Cheriton, David, Liu, Mike, Smith, Alan, Stankovic, Jack, Roby, Clyde, and Salasin, John, "WIS Operating System Specification," 1986.

[CHERI87a] Cheriton, David R., "Effective Use of Large RAM Diskless Workstations with the V Virtual Memory System," Computer Science Department, Stanford University, February 16, 1987.

[CHERI87b] Cheriton, David R., "UIO: A Uniform I/O System Interface," ACM Transactions on Computer Systems, Vol. 5, No. 1, pp.12-46, February 1987.

[CHERI88a] Cheriton, David R. and Timothy P. Mann, "Decentralizing a Global Naming Service for Improved Performance and Fault Tolerance," to appear in ACM Transactions on Computer Systems, 1988.

[CHERI88b] Cheriton, David R., "Exploiting Recursion to Simplify RPC Communication Architectures," Computer Science Department, Stanford University, Draft Paper, March 21, 1988.

[CHERI88c] Cheriton, David R., "The V Distributed System,"

Communications of the ACM, Vol. 31, No. 3, pp. 314-333, March 1988.

[CHERI88d] Cheriton, David R., "VMTP: Versatile Message Transaction Protocol," RFC 1045, SRI Network Information Center, February 1988.

[CHOU82] Chou, Timothy C.K, and Abraham, Jacob A., "Load Balancing in Distributed Systems," IEEE Transactions on Software Engineering, SE-8,4, pp. 401-412, July 1982.

[CLARK85] Clark, David D., "The Structuring of Systems Using Upcalls," Proceedings of the Tenth ACM Symposium on Operating Systems Principles, pp. 171-180, 1985.

[CLARK88] Clark, R. K., Kegley, R. B., Keleher, P. J., Maynard, D. P., Northcutt, J. D., Shipman, S. E. and Zimmerman, B. A., "An Example Real-Time Command and Control Application on Alpha," Archons Project Technical Report #88032, Department of Computer Science, Carnegie-Mellon University, March 1988.

[COOK85a] Cook, Robert P., "PHOENIX, A High Performance UNIX with an Emphasis on Dynamic Modification, Real-Time Response and Survivability," Submitted to Army Institute for Research in Management, Proposal No. CS-DOD/Army-3184-36, July 1985.

[COOK85b] Cook, Robert P., and Auletta, Richard J., "StarLite, A Visual Simulation Package for Software Prototyping" Technical Report, University of Virginia.

[COOPE87] Cooper and Draves, "C Threads," Computer Science Department, Carnegie Mellon University, Draft Paper, 2 March 1987.

[CORNHILL79] Cornhill, D.T.; Boebert, W.E., Implementation of the HXDP Executive, In Proceedings of COMPCON 79, pages 219-221, IEEE, February, 1979.

[COURT89] Courtright, J. and B. Haleen, "SDX Survey Summary"

[DASGU88] Dasgupta, Partha, Richard J. LeBlanc, and William F. Appelbe, "The Clouds Distributed Operating System: Functional Description, Implementation Details and Related Work," Proceedings of The 8th International Conference on Distributed Computing Systems, pp. 2-9, June 1988.

[DAVIE81] Davies, D.W., et al., Distributed Systems - Architecture and Implementation. Edited by B.W. Lampson, M. Pau, and H.J. Siegert. Springer-Verlag, New York, 1981.

[DAVIS87] Davis, Y.S., H.C. Younger, M.L. Lavin, "Command and Control Information Utility (CCIU)," prepared for INFOCOM '87.

[DEAN86] Dean, M.A., R.M. Sands, and R.E. Schantz., "Canonical Data Representation in the Cronus Distributed Operating System," Paper submitted to the ACM SIGCOMM '86 Symposium on Communication Architecture and Protocols, February 1986.

[DEAN87] Dean, Michael A., Richard M. Sands, and Richard E. Schantz, "Canonical Data Representation in the Cronus Distributed Operating System," Proceedings of the IEEE Infocom '87, pp. 814-819, March 1987.

[DEAN88] Dean, Mike, "Cronus, A Distributed Operating System: Ada Integration Investigation," Cronus Project Technical Report No. 7, Report No. 6797, BBN Laboratories Incorporated, April 1988.

[DHALL78] Dhall, S.K., and C.L. Liu "On a Real-Time Scheduling Problem," Operations Research, Vol. 26, No. 1, pp. 127-140, February 1978

[DIJKS80] Dijkstra E., Scholten C., "Termination Detection for Diffusing Computation," Inf. Proc. Letters, Vol 11, No. 1, pp. 1-4, August 1980.

[DIJKS83] Dijkstra E., Feijen W., Van Gasteren A., "Derivation of a Termination Detection Algorithm for Distributed Computation," Inf. Proc. Letters, Vol 16, pp. 217-219, June 1983.

[DRAVE88] Draves, Richard R., Michael B. Jones, and Mary R. Thompson, "MIG - The MACH Interface Generator," Computer Science Department, Carnegie Mellon University, Draft Paper, February 26, 1988.

[EFE82] Efe, K., "Heuristic Models of Task Assignment Scheduling in Distributed Systems," IEEE Computer, pp. 50-56, June 1982.

[ESPRI89] Esprit Systems Consulting Inc., "Structural Analysis for Real-Time System," P.O. Box 1486, WestChester, PA, 19380, January 30 - February 3, 1989.

[FINKEL83] Finkel, R.; Solomon, M.; DeWitt, D.; Landweber, L., The Charlotte Distributed Operating System: Part IV of the First Report on the Crystal Project., Technical Report, University of Wisconsin, 1983.

[FINLA87] Finlayson, Ross S. and David R. Cheriton, "Log Files: An Extended File Service Exploiting Write-Once Storage," Proceedings of the 11th Symposium on Operating System Principles, pp. 139-148, November 1987.

[FISHE86] Fisher, David A. and Weatherly, Richard M., "Issues in the Design of a Distributed Operating System for Ada," Computer, pp.38-47, May 1986.

[FRAIM83] Fraim, L.J., "SCOMP: A Solution to the MLS Problem," Computer, pp. 26-34, July 1983.

[FRIED83] Friedrich, G.R.; Eser, F.W., Management Units and Interprocess Communication of DINOS., Siemens Forsch.- and Entwicklungsber. (Germany) 12(1):21-27, January, 1983.

[GARCI83] Garcia-Molina, H., "Using Semantic Knowledge for Transaction Processing in a Distributed Database" ACM Transaction on Database Systems, Vol 8, No. 2, June 1983.

[GAREY75] Garey, M.R., and D.S. Johnson, "Complexity Results for Multiprocessor Scheduling Under Resource constraints," SIAM Journal of Computing, pp. 397-411, 1975.

[GAREY77] Garey, M.R. and D.S. Johnson "Two-Processors Scheduling with Start-times and Deadlines," SIAM Journal on Computing, Vol. 6, pp. 416-426, 1977.

[GATEF81] Gatefait, J.P.; Surleau, P.; Konrat, J.L., Execution Mechanisms for Administration Programs in the E10.S System., In IEE Fourth International Conference on Software Engineering for Telecommunication Switching Systems (Coventry England), pages 130-137., IEE, July 20-24,1981.

[GELER85] Gelernter, D., "Generating Communication in Linda," ACM Transaction on Computer Systems, January 1985.

[GESAL89] Gesalman, Paul and Barbara Haleen, "RSS/M Survey Summary"

[GIT86] The School of Information and Computer Science, Georgia Institute of Technology, "Effective Distributed Computing: A Reliable Object-Based Environment for Computer Science Research," A Proposal to the National Science Foundation's Co-ordinated Experimental Research Program, September 15, 1986.

[GLORIEUX81] Glorieux, A.M.; Rolin, P.; Sedillot, S., User Services Offered by the Application Protocol Implemented in SIRIUS-DELTA, In Networks from the User's Point of View, Proceedings of the IFIP TC-6 Working Conference COMNET '81 (Budapest, Hungary), pages 107-115, IFIP, May 11-15, 1981.

[GOODB82] Goodbody, R.L., "C3I System Engineering and Integration Concepts," NOSC Technical Document 538, San Diego, CA, July 1982.

[GRIEF86] Greif, I., R. Seliger and W. Weihl Atomic Data Abstractions in a Distributed Collaborative Editing System, (Extended Abstract) Conference Record of the thirteenth Symposium on Principles of Programming Languages, ACM SIGACT/SIGPLAN, St. Petersburg Beach, FL., January 1986.

[GURWI86] Gurwitz, R.F., M.A. Dean, and R.E. Schantz., "Programming Support in the Cronus Distributed Operating System." From Proceedings of the 6th International Conference on Distributed Computer Systems, IEEE Computer Society Press Washington DC, pp. 486-493, May 1986.

[HALEE89] Haleen, Barbara, "SDEX/44 Survey Summary"

[HALSTEAD80] Halstead, R.J. Jr; Ward, S.A., The MuNet: A Scalable Decentralized Architecture for Parallel Computation, In Proceedings of the Seventh Annual Symposium on Computer Architecture, pages 139-145, IEEE, 1980.

[HERLI87] Herlihy, M.P. and J.M. Wing, Avalon: Language Support for Reliable Distributed Systems. Proceedings of the 17th International Symposium on Fault-Tolerant Computing, July 1987

[HOOD86] Hood, P. and Grover, V., "Designing Real-Time Systems in Ada," SofTech, Inc., Waltham, MA Tech Report 1123-1, January 1986.

[HUAN 89] Huang, J., et al., "Experimental Evaluation of Real-Time Transaction Processing", COINS Technical Report 89-48, April 1989.

[JAHAN79] Jahanian F., and A.K. Mok, "Safety Analysis of Timing PRoperties in Real-Time Systems," IEEE Transactions on Software Engineering, SE-12(9) pp. 890-904, September 1986.

[JENSE78] Jensen, E.D., The Honeywell Experimental Distributed Processor--An Overview, IEEE Computer 11(1):28-38, January, 1978.

[JENSE81] Jensen, E.D., "Decentralized Control, "Distributed Systems: An Advanced Course, Springer-Verlag, 1981.

[JENSE85a] Jensen, E. Douglas, C. Douglass Locke, and Hideyuki Tokuda, "A Time-Driven Scheduling Model for Real-Time Operating Systems," Proceedings of IEEE Real-Time Systems Symposium, pp. 112-122, December 1985.

[JENSE85b] Jensen, E.D. et al Decentralized System Control, Technical Report RADC-TR-85-199, Carnegie Mellon University and Rome Air Development Center, April 1985.

[JENSE88a] Alpha Preview: A Briefing and Technology Demonstration for DoD. Archons Project Technical Report #88031, Department of Computer Science, Carnegie-Mellon University, March 1988.

[JENSE88b] Jensen, E. D., Northcutt, J. D., Clark, R. K., Shipman, S. E., Maynard, D. P. and Lindsay, D.C. The Alpha Operating System: An Overview. Archons Project Technical Report #88121, Department of Computer Science, Carnegie-Mellon University, December 1988.

[JENSE88c] Jensen, E.D., "Alpha: A Real-Time Decentralized

Operating System for Mission-Critical System Integration and Operation," Proc. Symposium on Integrated Computing Environments for Large, Complex Systems, University of Houston Research Institute for Computer and INformation Sciences, 1988.

[JENSE88d] Jensen, E.D., Test, J.A., Reynolds, F.D., Burke, E., Hanko, J.G. Alpha Release 2 Design Summary Report. Technical Report #88120, Kendall Square Research Corporation, September 1988.

[JENSE89] Jensen, E. D., Northcutt, J. D., Clark, R. K., Shipman, S. E., Reynolds, F. D., Maynard, D. P., and Loepere, K. L. Alpha: An Operating System for the Mission-Critical Integration and Operation of Large, Complex, Distributed Real-Time Systems 1989 Workshop on Operating Systems for Mission-Critical Computing, August 1989.

[JESSOP82] Jessop, W.H.; Noe, J.D.; Jacobson, D.M.; Baer, J.L.; Pu, C., The Eden Transaction-Based File System., In Proceedings of the Second Symposium on Reliability in Distributed Software and Database Systems (Pittsburgh PA), pages 163-169. IEEE,July 19-21, 1982.

[JONES79] Jones,A.K.; Chansler, R. J., Jr.; Durham, I; Schwans, K.; Vegdahl, S. R.; StarOS, a Multiprocessor Operating System for the Support of Task Forces, In Proceedings of the Seventh Symposium on Operating Systems Principles (SIGOPS), ACM, 1979.

[JONES86] "Mach and Matchmaker: Kernel and Language Support for Object-Oriented Distributed Systems," Proceedings of the 1st Annual ACM Conference on Object-Oriented Programming Systems, Languages, and Applications (OOPSLA), September 1986.

[KANAK87] Kanakia, Hemant (Electrical Engineering Department) and David R. Cheriton (Computer Science Department), "The VMP Network Adapter Board (NAB): High-Performance Network Communication for Multiprocessors," Stanford University, December 14, 1987.

[KANDL89] Kandlur, Dilip D. ,Daniel L. Kiskis, and Kang G. Shin (Department of EE and CS University of Michigan), "A Real-Time Operating System for HARTS" from the 1989 Workshop on Operating Systems for Mission Critical Computing, September 19-21, 1989.

[KERMA79] Kermani, P. and L. Kelinrock, "Virtual Cut-through: A New Computer Communication Switching Technique, Computer Networks, Vol. 3, pp. 267-286, 1979.

[KRISH87] Krishna, C.M., K.G. Shin, and I.S. Bhandari, "Processor Tradeoffs in Distributed Real-Time Systems," IEEE Trans. Comput., Vol. C-36, No. 9, pp. 1030-1040, September 1987.

[KRUEM89] Kruempel, K. and B. Haleen, "43RSS Survey Summary"

[LAMPO78] Lamport, L., "Time, Clocks, and Ordering of Events in a Distributed System," Communications of the ACM, Vol. 21, No. 7, pp. 558-565, July 1978.

[LAMPO82] Lamport, L., R. Shostak and M. Pease, "The Byzantine Generals Problem," ACM Trans. on Prog. Lang. and Systems, Vol. 4, No. 3, pp. 382-401, July 1982.

[LAMPO86] Lamport, L., "On Interprocess Communication parts I and II," Distributed Computing Vol. 1, No. 2, pp. 77-101, Springer-Verlag 1986.

[LANDW83] Landwehr, Carl E., "The Best Available Technologies for Computer Security," Computer, pp. 86-100, July 1983.

[LANTZ82] Lantz, K.A.; Gradischnig, K.D.; Feldman, J.A.; Rashid, R.F., Rochester's Intelligent Gateway, IEEE Computer 15(10): 54-68, October, 1982.

[LAU83] Lau, F.; Bei, J.; El-Bakoury, H.; Radia, S.; Tokuda, H.; Manning, E., Shoshin User's Guide, Technical Report, Computer Communications Networks Group, Institute for Computer Research, University of Waterloo, September, 1983.

[LAZOWSKA81] Lazowska, E.D.; Levy, H.M.; Almes, G. T.; Fischer, M.J.; Fowler, R.J.; Vestal, S.C., The Architecture of the Eden System., Operating Systems Review 15(5):148-159, December, 1981.

[LEBLA88] Leblanc, Thomas J. and Barton P. Miller, editors, "Summary of ACM Workshop on Parallel and Distributed Debugging," held May 5-6, 1988, University of Wisconsin, Madison, Wisconsin, ACM Operating Systems Review, Vol. 22, No. 4, pp. 7-19, October 1988.

[LEHOC86a] Lehoczky, J.P. and Sha, L., "Performance of Real-Time Bus Scheduling Algorithms", ACM Performance Evaluation Review, Special Issue, Vol 14, No. 1, May 1986.

[LEHOC86b] Lehoczky, John P., Hide Tokuda, Lui Sha, and Dennis Cornhill, "ART: An Advanced Real-Time Technology Project," Computer Science Department, Carnegie Mellon University, Draft Paper, November 28, 1986.

[LELANN81] LeLann, G., A Distributed System for Real-Time Transaction Processing, IEEE Computer 14(2):43-48, February, 1981.

[LEVI86] Levi, S., Plateau B., "A Distributed Algorithm for Deadlock and Termination Detection of Distributed Computations," University of Maryland Technical Report CS-TR-1750, University of Maryland, Department of Computer Science, December 1986.

[LINT79] Lint, Bernard, "Communication Issues in Parallel

Algorithms and Computers," Ph.D. Dissertation, Computer Science Department, University of Texas, May 1979.

[LISKO83] Lisko, B, and R. Scheiffler. Guardians and Actions: Linguistic Support for Robust Distributed Programs, ACM Transactions on Programming Languages and Systems, Vol. 53, July 1983.

[LISKO87] Liskov, B., D. Curtis, P. Johnson and R. Scheiffler. Implementation of Argus, Proceedings of the 11th ACM Symposium on Operating Systems Principles, November 1987.

[LIU73] Liu, C.L., and J.W. Layland, "Scheduling Algorithms for Multiprogramming in a Hard Real-Time Environment," Journal of the Association for Computing Machinery, Vol. 20, pp. 46-61, January 1973 .

[LIU82] Liu, M.T.; Tsay, D.P.; Lian, R.C., Design of a Network Operating System for the Distributed Double-Loop Computer Network (DDLCN), In Local Computer Networks, Proceedings of the IFIP TC 6 International In-Depth Symposium on Local Computer Networks (Florence, Italy), pages 225-248, IFIP, April 19-21, 1982.

[LOCKE84] Locke, C. Douglass, Decentralized Operating Systems A Survey, Carnegie-Mellon University, Pittsburgh, Pennsylvania, 1984

[LOCKE86] Locke, C. Douglass, Best-Effort Decision Making for Real-Time Scheduling, Ph.D. Dissertation, Carnegie Mellon University, 1986.

[LORIN80] Lorin, H., "Aspects of Distributed Computer Systems," John Wiley and Sons, New York, 1980.

[MA81] Ma P., Lee E., Tsuchiya M., "Design of Task Allocation Scheme for Time Critical Applications," IEEE Proceedings - Real Time Systems Symposium, Miami Beach FL December 1981.

[MAISON81] Maisonneuve, M.; Levy, J.P.; Konrat, J.L., E10.S Operating System for a Distributed Architecture., In IEE Fourth International Conference on Software Engineering for Telecommunication Switching Systems (Coventry, England), pages 124-129., IEE, July 20-24, 1981.

[MARZU85] Marzullo, K., Owicki S., "Maintaining the Time in a Distributed System," ACM Operating Systems Review, Vol 19, No. 3, pp. 44-54, July 1985.

[MILLE83] D.S. Miller, et al: "A Distributed Operating System for a Local Area Network." Proceedings of the Second Annual Phoenix Conference on Computers and Communications, IEEE Computer Society, 1983, pp. 281-288.

[MOK83] Mok A., "Fundamental Design Problems for the Hard Real Time Environment," MIT Ph.D. Dissertation, Cambridge, MA, May 1983.

[MOVAG84] Movaghar, A., and Meyer, J.F., "Performability Modeling with Stochastic Activity Networks," Proceedings of the Real-Time Systems Symposium, pp. 215-224, 1984.

[NGCR89a] NGCR OSSWG Reference Model, Version 1.03, December 6, 1989.

[NGCR89b] NGCR OSSWG Requirements Document, Version x.x, Date.

[NORTH87] Northcutt, J.D., "Mechanisms for Reliable Distributed Real Time Operating Systems - The Alpha Kernel," Perspectives in Computing, Academic Press, Vol. 16, 1987.

[NORTH88a] Northcutt, J. D. and Clark, R. K., "The Alpha Operating System: Programming Model," Archons Project Technical Report #88021, Department of Computer Science, Carnegie-Mellon University, February 1988.

[NORTH88b] Northcutt, J. D. and Shipman, S. E., "The Alpha Operating System: Program Maintenance Manual," Archons Project Technical Report #88123, Department of Computer Science, Carnegie-Mellon University, December 1988.

[NORTH88c] Northcutt, J. D. and Shipman, S. E., "The Alpha Operating System: Programming Utilities," Archons Project Technical Report #88041, Department of Computer Science, Carnegie-Mellon University, April 1988.

[NORTH88d] Northcutt, J. D., "The Alpha Distributed Computer System Testbed," Archons Project Technical Report #88033, Department of Computer Science, Carnegie-Mellon University, March 1988.

[NORTH88e] Northcutt, J. D., "The Alpha Operating System: Kernel Programmer's Interface Manual," Archons Project Technical Report #88111, Department of Computer Science, Carnegie-Mellon University, November 1988.

[NORTH88f] Northcutt, J. D., "The Alpha Operating System: Requirements and Rationale," Archons Project Technical Report #88011, Department of Computer Science, Carnegie-Mellon University, January 1988.

[NORTH88g] Northcutt, J. D., Clark, R. K., Shipman, S. E. and Lindsay, D. C., "The Alpha Operating System: System/Subsystem Specification," Archons Project Technical Report #88122, Department of Computer Science, Carnegie-Mellon University, December 1988.

[NORTH88h] Northcutt, J. D., Clark, R. K., Shipman, S. E., Maynard, D. P., Lindsay, D. C., Jensen, E. D., Smith, J. M., Kegley, R. B.,

Keleher and Zimmerman, B. A., "Alpha Preview: A Briefing and Technology Demonstration for DoD," Archons Project Technical Report #88031, Department of Computer Science, Carnegie-Mellon University, March 1988.

[OLSON83] Olson, R. A.; Kumar, B.; Shar, L.E., Messages and Multiprocessing in the ELXSI System 6400., In IEEE, Proceedings of the Spring 1983 COMPCON, pages 1-4,IEEE,1983.

[OSIa] DIS 7498-4 Open Systems Interconnection - Systems Management Overview

[OSIb] DIS 9072-1 Remote Operations, Part 1 Model, Notation and Service Definition

[OSIc] DIS 9506-1 Industrial Automation Systems - Systems Integration and Communications - Manufacturing Message Specification, Part1: Service Definition

[OSId] DIS 9594-1 The Directory, Part 1: Overview of Concepts, Models, and Services [OSIe] DP 10040 Open Systems Interconnection - Performance Management Working Document

[OSIf] IS 7498-4 Open Systems Interconnection - Basic Reference Model DIS 7498-3 Open Systems Interconnection - Basic Reference Model - PART 3: Naming and Addressing

[OSIg] IS 8471-1 File Transfer, Access, and Management (FTAM), Part 1: General Introduction

[OSIh] IS 8824 Specification of Abstract Syntax Notation 1 (ASN.1)

[PETER84] Peterson J., Silberschatz A., Operating System Concepts, Addison-Wesley Publishing Co., Reading, MA, July 1984.

[PITTS88] Pitts, David V. and Partha Dasgupta, "Object Memory and Storage Management in the Clouds Kernel," Proceedings of The 8th International Conference on Distributed Computing Systems, pp. 10-17, June 1988.

[POPEK79] Popek G., "Issues in Kernel Design," in Operating Systems: An Advanced Course, Bayer, R., Graham R., and Seegmuller G - editors, Springer-Verlag, Berlin, Germany, 1979.

[POPEK81] Popek, G.J., et. al., "LOCUS: A Network Transparent, High Reliability Distributed System," Proceedings Eighth Symposium of Operating Systems Principles, Pacific Grove, CA, December 1981.

[POSIX89] POSIX Realtime Extension for Portable Operating Systems, P1003.4/Draft 6, Draft 7, May 19, 1989

[PROJE89] Project Technology, Inc., "Object Oriented Systems

Analysis Information Models," 2560 Ninth Street, Suite 214, Berkeley, CA 94710, March 6-9, 1989.

[QUIRK85] Quirk W. (editor), Verification and Validation of Real Time Software, Springer-Verlag, Berlin, Germany, 1985.

[RAMAC88a] Ramachandran, U., M. Ahamad and M. Khalidi, Unifying Synchronization and Data Transfer in Maintaining Coherence of Distributed Shared Memory, Technical Report, GIT-ICS-88/23, School of Info. and Computer Science, Georgia Tech.

[RAMAC88b] Ramachandran, U., and Y.A. Khalidi, Memory Management Support for Object Invocation. Technical Report GIT-ICS-88/03. School of Information and Computer Science, Georgia Tech.

[RAMAM84] Ramamritham, K., and Stankovic, John A., "Dynamic Task Scheduling in Distributed Hard Real-Time Systems," Software, July 1984.

[RAMAM89] Ramamritham, Krithi, et al., "Overview of the Spring Project," Dept. of Computer and Information Sciences University of Massachusetts, January, 1989.

[RAMAM89-1] Ramamritham, Krithi, et al., "Efficient Scheduling Algorithms by Real-Time Multiprocessor Systems," COINS Technical Report 89-37, Dept. of Computer and Information, Science, University of Massachusetts

[RANDE82] Randell, B., "Structuring of Distributed Computing Systems," Technical Report, Newcastle upon Tyne University, 1982.

[RASHI81] Rashid, R.F. and G.G. Robertson. Accent: A Communication Oriented Network Operating System Kernel. Proc. of the Eighth Symposium on Operating Systems Principles, Operating Systems Review 15(5):pp 64-75, December 1981.

[RASHI87a] Rashid, Richard F., "From RIG to Accent to Mach: The Evolution of a Network Operating System," Computer Science Department, Carnegie Mellon University, August 28, 1987.

[RASHI87b] Rashid, Richard, et al., "Machine-Independent Virtual Memory Management for Paged Uniprocessor and Multiprocessor Architectures," Proceedings of the ACM Conference on Architectural Support for Programming Languages and Operating Systems, October 1987.

[RAZOU86] Razouk, R.R. T. Stewart and M. Wilson, "Measuring Operating System Performance on Modern Microprocessors," Performance 86, pp. 193-202, 1986.

[READY86] Ready, J.F., "VRTX: A Real-Time Operating System for Embedded Microprocessor Applications," IEEE Micro, pp. 8-17,

August, 1986.

[REYNO88a] Reynolds, F.D., Hanko, J.G., Jensen, E.D. Alpha Release 2 Preliminary System/Subsystem Description. Technical Report #88122, Concurrent Computer Corporation, December 1988.

[REYNO88b] Reynolds, F.D., Hanko, J.G., Test, J.A., Burke, E., Jensen, E.D. Alpha Release 2 Kernel Interface Specification. Technical Report #88121, Concurrent Computer Corporation, December 1988.

[RICAR81] Ricart G., Agrawala A., "An Optimal Algorithm for Mutual Exclusion in Computer Network," Communication of the ACM, Vol. 23, No. 1, pp. 9-17, January 1981

[RUSHB83] Rushby, J.M., and Randell B., "A Distributed Secure System," Computer, pp. 55-67, July 1983.

[SALTZ79] Saltzer J., "Naming and Binding of Objects," in Operating Systems: An Advanced Course, Bayer, R., Graham, R., and Seegmuller G - editors, Springer-Verlag, Berlin, Germany, 1979.

[SANSO86] Sansom, Robert D., Daniel P. Julin, and Richard F. Rashid, "Extending a Capability Based System into a Network Environment," Technical Report CMU-CS-86-115, Computer Science Department, Carnegie Mellon University, April 24, 1986.

[SCHAN85] Schantz, R., et al., "CRONUS, A Distributed Operating System: Phase 1 Final Report," Report No. 5885, BBN Laboratories Incorporated, January 1985.

[SCHAN86a] Schantz, R., R.H. Thomas, and G. Bono "The Architecture of the Cronus Distributed Operating System," From: Proceedings of the 6th International Conference on Distributed Computer Systems, IEEE Computer Society Press, Washington, DC, pp. 250-259, May 1986.

[SCHAN86b] Schantz, R., et al., "CRONUS, A Distributed Operating System: Cronus DOS Implementation, Final Report," Report No. 6183, BBN Laboratories Incorporated, March 1986.

[SCHMID82] Schmidtke,F. E., A Communication Oriented Operating System Kernel for a Fully Distributed Architecture, In Pathways to the Information Society. Proceedings of the Sixth International Conference on Computer Communication (London, England), pages 757-762., International Council of Computer Communication, September 7-10, 1982.

[SCHMID83] Schmidtke, F. E., Operating System for an Optical-Bus Local Network., Siemens Forsch.- and Entwicklungsber. (Germany) 12(1):16-20, January, 1983.

[SCHWA87] Schwan, K.P., Gopinath and T. Bo, "CHAOS: Kernel Support for Objects in the Real-Time Domain,"IEEE Transactions on Computers, pp. 904-916, August 1987.

[SEDILLOT80] Sedillot, S., and Sergeant, G., The Consistency and Execution Control Systems for a Distributed Data Base in SIRIUS-DELTA, Paper proposed to IFIP 80 Congress.

[SELVI86] Selvin, Manny, "Distributed Processing Relative to Distributed Command, Control, Communications (DC3)." MITRE Report prepared for Joint Directors of Laboratories C3 Research and Technology Program, January 1986.

[SHANK84] Shankar A.U., Lam S.S, "Time Dependent Communication Protocols," Tutorial: Principles of Communication and Networking Protocols, S.S. Lam (ed), IEEE Computer Society, 1984.

[SHANK86] Shankar, A.U., Lam, S.S., "Construction of Sliding Window Protocols," CS-TR-1647 Computer Science Department, University of Maryland, February 1986.

[SPECT88] Spector, Alfred Z. and Kathryn R. Swedlow, editors, Guide to the Camelot Distributed Transaction Facility: Release 1, Computer Science Department, Mach/Camelot, Carnegie Mellon University, Draft of February 4, 1988.

[STANK85] Stankovic, John A., Ramamrithm, Krithivasan and Kohler, Walter H., "A Review of Current Research and Critical Issues in Distributed System Software," Distributed Processing Technical Committee Newsletter, pp. 14-47, March 1985.

[STANK87] Stankovic, J.A. and K. Ramaamritham, Proceedings of the Real-Time Systems Symposium, pp. 146-157, December 1987.

[STAN88] Stanhovic, John A., et al., "The Design of the Spring Kernel," COINS Technical Report 88-85, Dept. of Computer an Information Service, University of Massachusetts.

[STAN 88-1] Stanhovic, John A., et al., "The Spring Kernel: A New Paradigm for Real-Time Operating System," COINS Technical Report 88-97, Dept. of Computer and Information Service, University of Massachusetts, November, 1988.

[STARK85] Stark, Gene, "Foundations of a Theory of Specification for Distributed Systems," Ph.D. Dissertation, Massachusetts Institute of Technology, 1985.

[STONE77] Stone, Harold, "Multi-processor Scheduling with the Aid of Network Flow Algorithms," IEEE Transactions on Software Engineering, SE-3, pp. 85-93, January 1977.

[STONE81] Stonebraker, Michael, "Operating System Support for

Database Management," CACM, pp. 412-418, July 1981.

[TANEN81] Tanenbaum, A.S. and S.J. Mullender. An Overview of the Amoeba Distributed Operating System. Operating Systems Review, Vol. 13, No. 3, pp. 51-64, July 1981.

[TANEN85] Tanenbaum, Von Renesse, R., "Distributed Operating Systems," ACM Computing Surveys, Vol 17, No. 4, pp. 419-470, December 1985.

[TCSEC85] "Department of Defense Trusted Computer System Evaluation Criteria," National Computer Security Center, DoD 5200.28-STD, December 1985.

[TEVAN87] Tevanian, Avadis, Jr., Architecture-Independent Virtual Memory Management for Parallel and Distributed Environments: The Mach Approach, Ph.D. Thesis, Technical Report CMU-CS-88-106, Computer Science Department, Carnegie Mellon University, December 1987.

[THEIM85] Theimer, Marvin M., Keith A. Lantz, and David R. Cheriton, "Preemptable Remote Execution Facilities for the V-System," Proceedings of the 10th Symposium on Operating System Principles, December 1985.

[TIS88] Trusted Information Systems, Inc., "Trusted Mach Presentation," Ellicott City, Maryland, December 7, 1988.

[TOKUDA83] Tokuda, Hideyuki; Radia S.R.; Manning, E.G., Shoshin OS: a Message-based Operating System for a Distributed Software Testbed, In Proceedings of the Sixteenth Hawaii International Conference on System Sciences, 1983 (Honolulu HI), pages 329-338, University of Hawaii, University of Southwestern Louisiana, January 5-7, 1983.

[TOKUD87] Tokuda, Hideyuki, James W. Wendorf, and Huay-Yong Wang, "Implementation of a Time-Driven Scheduler for Real- Time Operating Systems," IEEE 8th Real-Time Systems Symposium, December 1987.

[TOKUD88a] Tokuda, Hideyuki, Makoto Kotera, and Clifford W. Mercer, "A Real-Time Monitor for a Distributed Real- Time Operating System," ACM SIGOPS/SIGPLAN Workshop on Distributed/Parallel Debugging.

[TOKUD88b] Tokuda, Hideyuki, and Makoto Kotera, "Scheduler 1-2-3: An Interactive Schedulability Analyzer for Real-Time Systems," Computer Science Department, Carnegie Mellon University, February 15, 1988.

[TOKUD89] Tokuda, Hideyuki, Mercer, Clifford, W., and Ishikawa, Yutaka, "The ARTS Distributed Real-Time Kernel and its Toolset," Carnegie Mellon University.

[TOPOR84] Topor R., "Termination Detection for Distributed Computation," Inf. Proc. Letters, Vol. 18, pp. 33-36, January 1984.

[TRIGG81] Trigg, R., Software on ZMOB: An Object-Oriented Approach, In 1981 IEEE Computer Society Workshop on Computer Architecture for Pattern Analysis and Image Database Management (Hot Springs VA), pages 133-140, IEEE, November 11-13, 1981.

[TRULL88] Trull, J. E., Northcutt, J. D., Clark, R. K., Shipman, S. E. and Lindsay, D. C. An Evaluation of Alpha Real-Time Scheduling Policies. Archons Project Technical Report #88123, Department of Computer Science, Carnegie-Mellon University, December 1988.

[TSAY81] Tsay, D.P.; Liu, M.T., MIKE: A Network Operating System for the Distributed Double-Loop Computer Network (DDLCN), In Proceedings of COMPSAC 81m IEEE Computer Society's Fifth International Computer Software and Applications Conference (Chicago, IL), pages 388-402, IEEE, November 16-20, 1981.

[VINTE87] Vinter, Stephen T., et al., "The Cronus Distributed DBMS Project: Functional Description," Report No. 6660, BBN Laboratories Incorporated, October 1987.

[VINTE88] Vinter, Stephen T., et al., "The Cronus Distributed DBMS Project: Program Specification," Report No. 6854, BBN Laboratories Incorporated, June 1988.

[WALKE83] Walker, Bruce, Popek Gerald, English Robert, Kline Charles, and Thiel, Greg., "The LOCUS Distributed Operating System," In Proceedings of the Ninth ACM Symposium on Operating Systems Principles, October 10-13, 1983.

[WEIHL83] Weihl, W. and B. Liskov, Specification and Implementation of Resilient Atomic Data Types, Symposium on Programming Language Issues in Software Systems, June 1983.

[WIRTH77] Wirth, N., "Toward A Discipline of Real Timing Programming," Communications of the ACM, Vol 20, No. 8, pp. 577-583, August 1977.

[WOOD89] Wood P.C., "SPRITE Survey Summary"

[WULF74] Wulf, W.A. and others, HYDRA: The Kernel of a Multiprocessor Operating System, Communications of the ACM, Vol. 17, No. 6, June 1974.

[YEE88] Yee, Bennet S., J.D. Tygar, Alfred Z. Spector, "Strongbox: A Self-Securing Protection System for Distributed Programs," Technical Report CMU-CS-87-184, Computer Science Department,

Carnegie Mellon University, January 4, 1988.

[YOUNG87] Young, Michael, et al., "The Duality of Memory and Communication in the Implementation of a Multiprocessor Operating System," Proceedings of the 11th Symposium on Operating Systems Principles, November 1987.

[YUAN87] Yuan, Xiaoping, Tripathi, Satish, and Agrawala, Ashok, "Scheduling in Real Time Distributed Systems - A Review" University of Maryland, UMIACS-TR-87-62, CS-TR-1955, December 1987.

Next Generation Computer Resources

Operating System Interface Standard Requirements

VERSION 2.0
21 December 1989

TABLE OF CONTENTS

1 INTRODUCTION

1.1 Scope

This document provides the Department of the Navy's requirements for the definition and specification of an Operating System Interface for the Next Generation Computer Resources (NGCR).

1.2 Terminology

Precise and consistent use of terms has been attempted throughout the document.

Potentially ambiguous terms used in the document are defined in the Glossary for the NGCR. Some definitions tailored to the context of this document are provided in the sections of the document where they are used.

Additionally, the following verbs and verb phrases are used throughout the document to indicate where and to what degree individual constraints apply. Any sentence not containing one of the following verbs or verb phrases is a definition, explanation or comment.

"PROCESS" indicates an operating system scheduiable entity. Other terms shall be qualified by a specific programming language, (i.e., Ada task, Pascal procedure, "C" program, etc.)

"SHALL PROVIDE" indicates a requirement for the operating system interface to provide interface(s) with prescribed capabilities.

"SHALL SUPPORT" indicates a requirement for the operating system interface to provide interface(s) with prescribed capabilities or for operating system interface definers to demonstrate that the capability can be constructed from operating system interfaces.

"SHOULD PROVIDE" indicates that the requirement is a desired goal of the OSIF.

2 REQUIREMENTS

2.1 GENERAL REQUIREMENTS

2.1.1 Scope

2.1.1.1 Definition

The OSIF shall provide interfaces sufficient to support a wide range of Navy target applications.

2.1.1.2 Metric

2.1.1.3 Rationale

It is intended that the OSIF will be used by applications from missile guidance systems to large command and control systems to completely integrated platforms. This range of target applications is very demanding, and it is known that there is no one operating system that can satisfy all possible application systems. But the goal of

will take highest priority in determining the appropriate features of the OSIF.

2.1.2 Design Objective

2.1.2.1 Definition

The OSIF should provide interfaces sufficient to promote compatibility, interoperability, transportability, and reusability between applications and maintainability of applications.

2.1.2.2 Metric

2.1.2.3 Rationale

This requirement addresses the reasoning behind the development of the OSIF. These are the qualities which are desired in applications and which can be promoted by the operating system interface. "Interoperability" is the ability of two applications to share data. "Transportability" is the ability to move an application from one implementation of the OSIF to another with minimal changes to the source code. "Reusability" is the ability to reuse portions of one application's source code or other pertinent aspects (e.g., design, tests) in the generation of another application. "Compatibility" is the general ability of two applications to coordinate with one another in their operation, even if they were not originally designed to do so. "Maintainability" addresses the qualities which improve the ability to maintain the application.

2.1.3 Basic Services

2.1.3.1 Definition

The OSIF should provide simple-to-use mechanisms for achieving common, simple actions. Facilities which support less frequently used features should be given secondary consideration.

2.1.3.2 Metric

2.1.3.3 Rationale

The OSIF should be understandable and usable. Thus this requirement encourages the selection of a set of interfaces where the frequently used ones (at compilation time) are simple to use, possibly at the expense of less frequently used facilities being more difficult to invoke. This requirement also suggests concentration on supporting those application actions that are likely to have the broadest utility.

2.1.4 Architecture Independence

2.1.4.1 Definition

The OSIF shall be machine-independent and implementation-independent. The OSIF shall be implementable on a wide variety of processors, configurations and architectures.

2.1.4.2 Metric

2.1.4.3 Rationale

The OSIF must depend on no properties of specific computers and on no properties of specific implementations. The features should also be chosen to have a simple and efficient implementation in many machines and hardware architectures and configurations (including distributed configurations). Transportability can only be achieved where the OSIF itself can be implemented on a wide range of machines without revealing or relying on machine or implementation dependencies.

2.1.5 Modularity

2.1.5.1 Definition

The OSIF should be partitioned such that the partitions can be understood independently.

2.1.5.2 Metric

2.1.5.3 Rationale

This criterion promotes understandability and permits application writers to employ a subset of the OSIF, which will be important for many applications. It should be noted that there can be multiple versions of some partitions within the standard. Independent understanding implies that there should be no undocumented dependencies between partitions.

2.1.6 Extensibility

2.1.6.1 Definition

The OSIF should facilitate development and use of extensions of the OSIF; e.g., OSIF interfaces should be composable so that they can be combined to create new interfaces and facilities, or it should be possible to add new interfaces for new functions.

2.1.6.2 Metric

2.1.6.3 Rationale

The state-of-the-art and the state-of-the-practice in computer technology are rapidly changing. It is impossible to fully list all interfaces that will be required in all future application domains. Therefore, the list of specialized interfaces must be extensible.

2.1.7 Uniformity

2.1.7.1 Definition

The OSIF should be based upon a consistent set of unifying well-defined conceptual models. all OSIF features should uniformly address aspects such as status return, exceptional conditions, parameter types, and options.

2.1.7.2 Metric

2.1.7.3 Rationale

The design of the OSIF should minimize the number of underlying concepts and unifying principles. A unifying

principle is a model which unifies (a subset of) the interfaces. It should have few special cases and should consist of features that are individually simple. These models should also be consistent with one another. However, these objectives are not to be pursued to the extreme of providing inconvenient mechanisms for the expression of some common, reasonable actions.

2.1.8 Completeness

2.1.8.1 Definition

The OSIF should provide a complete set of facilities for all elements of its underlying conceptual models.

2.1.8.2 Metric

2.1.8.3 Rationale

Since one of the major goals of the OSIF is transportability, it must provide a sufficient set of facilities to support applications so they do not have to utilize facilities outside of the OSIF. While it is desirable that the OSIF be complete and provide all facilities for applications, a requirement that mandates all facilities for all applications is recognized to be unachievable in practice. The goal for OSIF should be to optimize the degree of completeness, compromising between all possible facilities and those that can be implemented widely. The things within the conceptual models of the OSIF can, in all probability, only be manipulated by the OSIF interfaces. Hence all desired manipulations must be catered for by the OSIF. Simple examples are:

- if there is a facility to create something, then there should also be a facility to delete it
- if there is a facility to set a value, then there should also be a facility to examine it.

2.1.9 Language Independence

2.1.9.1 Definition

The OSIF should include a clear description of its interfaces which is independent of any particular programming language binding.

2.1.9.2 Metric

2.1.9.3 Rationale

Although Ada is the language of primary interest to the Navy, other languages will be important as well. For example, the NGCR objective of being able to purchase commercial-off-the-shelf components often involves such popular languages as 'C', and Navy systems in the 21st century might well incorporate intelligent subsystems which are written in popular artificial intelligence languages. The best way to evolve a standard which can withstand such demands is to develop the services in a language-independent way, thus allowing the development of any number of compatible language bindings. In order to achieve this, the basic description must be complete, consistent, unambiguous, and abstracted away from the details of any particular programming languages, but capable of accommodating a variety of languages.

2.1.10 Ada Language Binding Syntax

2.1.10.1 Definition

The OSIF shall have an Ada language binding consistent with the language independent model. The OSIF Ada binding syntax shall be expressed as Ada package specifications, as defined by the Ada standard (MIL-STD/ANSI-1815A). It shall provide a fully-documented interface from Ada to all operating system facilities for which there is no appropriate Ada language construct.

2.1.10.2 Metric

2.1.10.3 Rationale

The interface should be fully specified in Ada (possibly in addition to other languages) to prevent ambiguities from arising concerning how the specification language maps to Ada. All ambiguities in specification will result in reduced portability. There will be many interfaces which provide facilities that are not directly supported by the Ada language that will be necessary for an application to have access to. It must be easy for the Ada programmer to gain access to these facilities.

2.1.11 Other Language Binding Syntax

2.1.11.1 Definition

The OSIF should have a variety of language bindings, consistent with the language independent model. The syntax of each shall be presented in a manner consistent with good practice for that language.

2.1.11.2 Metric

2.1.11.3 Rationale

A number of language bindings other than Ada will also be desirable. Each should exhibit good style for that language and be presented according to the accepted standard for the language, if such a standard

2.1.12 Language Binding Syntax Uniformity

2.1.12.1 Definition

Each OSIF language binding should employ uniform syntactic conventions and should not provide several notations for the same concept.

2.1.12.2 Metric

2.1.12.3 Rationale

OSIF language binding syntax issues (including, at least, limits on name lengths, abbreviation styles, other naming conventions, and relative ordering of input and output parameters) should be resolved in a uniform and integrated manner for the whole OSIF language binding. Understandability and usability of the OSIF are the intents of this criterion. Users should not have to unnecessarily learn different syntactic approaches for using different OSIF features. The use of several notations for the same concept leads to confusion, and non-uniformity is a recipe for errors in use and difficulty in production of applications.

2.1.13 Language Binding Syntax Name Selection

2.1.13.1 Definition

The OSIF language bindings should avoid coining new words (literals or identifiers) and should avoid using words in an unconventional sense. Identifiers (variable names) defined by the OSIF language bindings should be natural-language words or industry-accepted terms whenever possible. The language bindings should define identifiers that are visually distinct and not easily confused. The language bindings should use the same name everywhere in the interface set, and not its possible synonyms, when the same meaning is intended.

2.1.13.2 Metric

2.1.13.3 Rationale

Understandability of the OSIF specification is the intent of this criterion.

2.1.14 Syntactic Pragmatics

2.1.14.1 Definition

The OSIF language bindings should impose only those restrictive rules or constraints required to support the design objectives (Refer to Section 2.1.2)

2.1.14.2 Metric

2.1.14.3 Rationale

Although it would be ideal if no such restrictions were required, practical considerations dictate that some limits will exist in all implementations. Where necessary to support the design objectives, such restrictions should be clearly articulated by the OSIF specification. If they are not necessary to support the design objectives (e.g., if they are present for the convenience of the OSIF or its implementers), then they are not desirable in the OSIF.

2.1.15 General Semantics

2.1.15.1 Definition

The OSIF should be completely and unambiguously defined. The specification of semantics should be both precise and understandable. The semantic specification of each OSIF interface shall include a precise statement of assumptions (including execution-time preconditions for calls), effects on global data and services, and interactions with other interfaces.

2.1.15.2 Metric

2.1.15.3 Rationale

This is a call for adequate, usable documentation. It is critical for the ability of independent vendors to develop implementations which can be used readily, both alone and together. Note that the requirement does not prescribe the degree of formality of the language to be used for specifying the OSIF semantics, admitting options from free-form English (as long as it is complete and precise) to a formal semantics specification approach.

2.1.16 Semantic Consistency

2.1.16.1 Definition

The description of OSIF semantics should use the same word or phrase everywhere, and not its possible synonyms, when the same meaning is intended.

2.1.16.2 Metric

2.1.16.3 Rationale

The use of synonyms, while desirable in novels, has no place in technical documents such as this and only lead to confusion.

2.1.17 Error Conditions

2.1.17.1 Definition

The OSIF language bindings shall employ appropriate mechanisms to report exceptional situations that arise in the execution of OSIF facilities. The OSIF specifications shall include error conditions for all situations that violate the preconditions specified for the OSIF interface. The OSIF specification shall define error conditions that cover all violations of implementation-defined restrictions.

2.1.17.2 Metric

2.1.17.3 Rationale

This requirement demands that the OSIF provide the means for the implementation to inform the user (i.e., an application) whenever an operation does not complete normally; this allows the application to take appropriate action. The use of the Ada exception mechanism is expected for the Ada language binding. However, use of the exception mechanism does not preclude the utilization of status return values by an Ada binding to provide supporting diagnostic information. Thus it would be possible to have a single exception representing a number of different (but related) error conditions, with the identification of the specific error condition via some status parameter.

2.1.18 Semantic Cohesiveness

2.1.18.1 Definition

Each OSIF interface should provide only one function.

2.1.18.2 Metric

2.1.18.3 Rationale

This criterion is a statement of the basic principle of software cohesion. It should be interpreted as making undesirable an interface design with a small number of entry points, each entry point delivering a range of dissimilar services, with particular services being selected dynamically by the value of one or more of the parameters. Such a design would almost certainly not provide simple-to-use interfaces for performing simple

common actions. While there can always be instances argued where one person views as several (sub)functions what someone else views as an atomic function, this should be the exception rather than the rule. The overloading of sub-program names allowing the selection of different sub-programs depending on the types of the parameters is acceptable. Such an approach has the advantage of reducing the number of unique sub-program names within the OSIF. When used to extreme, overloading can lead to a reduction in the clarity of the OSIF and should be used with care.

2.1.19 Semantic Pragmatics

2.1.19.1 Definition

The OSIF specification shall enumerate all aspects of the meanings of OSIF interfaces and facilities which must be defined by OSIF implementers. OSIF implementers will be required to provide the complete specifications for these implementation-defined semantics.

2.1.19.2 Metric

2.1.19.3 Rationale

This calls for the equivalent of Appendix F of MIL/ANSI-STD-1815A, containing a list of specific sections or aspects of the OSIF specification where implementation dependencies (presumably due to machine dependencies) are allowed to affect OSIF semantics. This list needs to be accompanied by a statement that no other implementation dependencies are allowed other than those listed, and that each OSIF implementation must include documentation stating the implementation characteristics for each item on the list.

2.1.20 Reaction to Blocking Services

2.1.20.1 Definition

The OSIF shall define what happens if a process calls on a service and that service cannot be completed in a timely manner.

2.1.20.2 Metric

2.1.20.3 Rationale

An application process may request an OS service which can be initiated immediately, but will not be completed until some later time. Examples would be delay services, I/O services on a device, file I/O services when system buffers are empty or full, and synchronization services. In these circumstances, the OSIF must define the effect on the requesting process and the overall effect on the scheduling of the requesting CPU. Many applications will require blocking of the requesting process, to allow other processes to use this CPU as defined by the scheduling algorithm; other applications may require that the requesting process continue to execute in parallel with the requested service (possibly being notified when the service is complete) or be notified that the service cannot be provided immediately (the application would repeat the request later). Busy wait within the OS is generally not an acceptable approach.

2.1.21 Bounded Operating Systems Services Times and Context Switching

2.1.21.1 Definition

The OSIF shall support the prediction of operating system service completion times. The OSIF shall be implementable such that these service times are bounded. The OSIF implementer will be required to document the service times.

2.1.21.2 Metric

2.1.21.3 Rationale

In order for a scheduler to predict performance with any degree of precision it is necessary to have a bound on the time that it can be expected to take for a requested operating system service to complete and to have access to the information necessary for the prediction of completion times. It is also important that the OSIF be implementable such that these service times are bounded and can be known for a given implementation.

2.1.22 Configurability

2.1.22.1 Definition

The OSIF shall be implementable such that application projects have the ability to configure the implementation to be optimal for the specific application.

2.1.22.2 Metric

2.1.22.3 Rationale

In certain situations, the user of an operating system will want to emphasize certain aspects of the operating system which in other situations the user (or another user) may wish to de-emphasize. Toward that end, the user must be able to configure the OSIF implementation and, for example, to select from a variety of scheduling options and synchronization mechanisms. There must be no penalty in space or time for services not used, and optimization for certain patterns of usage must be available.

2.1.23 Transaction Scheduling Information

2.1.23.1 Definition

The OSIF shall provide the ability for a process to specify its response requirements for services.

2.1.23.2 Metric

2.1.23.3 Rationale

Many forms of application processing, in particular database management transaction processing (for hard real-time systems), need to be scheduled so as to meet the response requirements of all transactions as well as the response requirement of all the application tasks co-resident with it. This service is not necessarily provided simply by the ability to dynamically set priorities.

2.1.24 Access Control

2.1.24.1 Definition

The OSIF shall provide a mechanism to allow only certain subjects (i.e., processes) to make use of particular objects (e.g., files, devices). That is, access to certain objects may be limited to only certain application software entities.

2.1.24.2 Metric

2.1.24.3 Rationale

A means must be provided to limit the access to certain system objects (e.g., files, devices or ports) to only those software entities with sufficient privileges. A mechanism which achieves this on a limited, predefined basis for only some objects would not completely fulfill this requirement.

2.1.25 Transparency

2.1.25.1 Definition

The OSIF shall provide for the identification of and access to processes and data, irrespective of their physical location.

2.1.25.2 Metric

2.1.25.3 Rationale

For many applications it will be necessary (or at least desirable) to be able to access processes and data via logical names rather than via physical location. Note that this requirement does not preclude other access methods which do relate to physical location.

2.1.26 Resilience

2.1.26.1 Definition

The OSIF shall be implementable so that, when physical resources are lost, OSIF facilities which do not depend on these resources may continue to be used.

2.1.26.2 Metric

2.1.26.3 Rationale

This requirement is intended to preclude the design of an OSIF such that the operation of an implementation must be dependent on the availability of the complete set of resources.

2.1.27 Network Partition

2.1.27.1 Definition

The OSIF shall be implementable so that partitions of the set (e.g., network) of physical resources can usefully work in isolation and the partitions may be rejoined after recovery.

2.1.27.2 Metric

2.1.27.3 Rationale

This requirement is primarily concerned with the support for an OSIF implementation which is based on a distributed architecture. An OSIF implementation, based on a distributed network of processors and other physical resources, which could only operate when all processors and interconnections were available would, particularly in a warfare environment, be a severe disadvantage. Indeed, this requirement should be read to demand that if a network fragments, then each segment which has sufficient resources to continue operation should do so. The OSIF must also facilitate subsequent reintegration of the network allowing the partial fragments to be recombined. The OSIF should, in particular, not be defined such that the only possible distributed implementation would be one that depended upon a single system- wide resource for its operation.

2.1.28 Reference

2.1.28.1 Definition

The OSIF shall provide a means to refer to distinct physical resources (e.g., computational, storage) that are used to implement specific OSIF facilities.

2.1.28.2 Metric

2.1.28.3 Rationale

While physical distribution may be transparent to most applications, there are circumstances under which specific parts of applications (most notably those concerned with system status and fault tolerance) may require or have knowledge about the allocation of application components to equipment in the underlying computer configuration. This requirement states that there must be a means for applications to be built to take advantage of such knowledge. Note that these capabilities will always be optional OSIF features in the sense that applications may choose never to reference physical resources and always to leave their mappings (and even the possibility of distributed implementations) up to OSIF implementers. Application writers should also be aware of possible sacrifices in transportability when using such features.

2.1.29 Reallocation

2.1.29.1 Definition

The OSIF shall provide a means to control (or influence) the manner in which the physical resources are associated with specific OSIF facilities.

2.1.29.2 Metric

2.1.29.3 Rationale

Certain applications may require particular operations to dynamically control the allocation of application components to underlying equipment in distributed configurations. This requirement states that there must be a means for applications to be built with such capability. Note that these capabilities will always be optional OSIF features in the sense that applications may choose never to reference physical resources and always to leave their mappings (and even the possibility of distributed implementations) up to OSIF implementers. Application writers should also be aware of possible sacrifices in transportability when using such features.

2.2 ARCHITECTURE DEPENDENT INTERFACES

2.2.1 Non-NGCR System Interfaces

2.2.1.1 Definition

The OSIF shall support non-NGCR-based systems by providing a subset of its services to those systems. As a minimum this subset shall include:

- Download, initialize, start, and stop
- Ability to share resources, particularly peripheral devices
- Process-to-process message communication
- Ability to pass operational status information

2.2.1.2 Metric

2.2.1.3 Rationale

The Navy has a large investment in existing non-NGCR-based systems. These systems will continue to be in use for years to come and are likely to need to interface to some degree with NGCR-based systems. Additionally they may need a method to gracefully transition to NGCR-based systems. The non-NGCR-based systems for their part will be required to change to accommodate the interface subset.

2.3 CAPABILITY AND SECURITY INTERFACES

2.3.1 Audit Data Storage

2.3.1.1 Definition

The OSIF shall support the storage and maintainability of audit data.

2.3.1.2 Metric

2.3.1.3 Rationale

The TCSEC (section 4.1.2.2) requires storage and maintenance of audit data for all systems at level C2 and above. This requirement does not specify what events are recorded in the log, which is implementation specific.

2.3.2 Audit Generation

2.3.2.1 Definition

The OSIF shall support generation of audit records.

2.3.2.2 Metric

2.3.2.3 Rationale

The TCSEC (section 4.1.2.2) requires the capability to generate audit records for all systems at level C2 and above. This requirement does not specify whether any privileges are required to use this feature.

2.3.3 Audit Record Contents

2.3.3.1 Definition

The OSIF shall support generation of audit records which uniquely identify the subject, event, and object being operated upon.

2.3.3.2 Metric

2.3.3.3 Rationale

The TCSEC (section 4.1.2.2) requires that data in the audit trail contain specific information which is used to identify the subject (e.g., process and user ID), object (e.g., data file), and the event which caused the audit record to be generated.

2.3.4 Audit Data Manipulation

2.3.4.1 Definition

The OSIF shall support the manipulation of audit data.

2.3.4.2 Metric

2.3.4.3 Rationale

The TCSEC (section 4.1.2.2) requires facilities to manipulate audit data for all systems at level C2 and above. Such facilities may include audit data analyzers and report generators.

2.3.5 Device Labels

2.3.5.1 Definition

The OSIF shall support the assignment of minimum and maximum security levels to all devices.

2.3.5.2 Metric

2.3.5.3 Rationale

The TCSEC (section 4.1.1.3.4) requires that all devices have minimum and maximum security levels for all systems at level B2 or above. This requirement includes both physical devices (e.g., disks, terminals) and logical devices (e.g., interprocess communication).

2.3.6 Basic DAC

2.3.6.1 Definition

The OSIF shall support a mechanism for the enforcement of discretionary access control (DAC) based on users and groups.

2.3.6.2 Metric

2.3.6.3 Rationale

The TCSEC (section 4.1.1.1) requires that minimal DAC facilities (for level C2) include enforcement based on a user and group mechanism. or levels B3 and above, the requirement is for access Control Lists (ACLs).

2.3.7 DAC Inclusion/Exclusion

2.3.7.1 Definition

The OSIF shall support the manipulation of access rights to specifically include or exclude access based on users or groups.

2.3.7.2 Metric

2.3.7.3 Rationale

The TCSEC (section 4.1.1.1) requires that the DAC mechanism specifically allow for inclusion based on individual users or groups for levels C2 and above. For levels B3 and above, the TCSEC also requires that the DAC mechanism allow for exclusion based on individual users or groups.

2.3.8 DAC Propagation

2.3.8.1 Definition

The OSIF shall provide controls to limit propagation of access rights.

2.3.8.2 Metric

2.3.8.3 Rationale

The TCSEC (section 4.1.1.1) requires that DAC rights granted to a user or group not be propagated to another user or group. In general, this is interpreted to mean that only the owner of an object (or a privileged user) may change the DAC on that object. This facility is required for security levels C2 and above.

2.3.9 Labelling of Export Channels

2.3.9.1 Definition

The OSIF shall support the restriction of the set of labels to be exported over each export channel.

2.3.9.2 Metric

2.3.9.3 Rationale

The TCSEC (section 4.1.1.3) requires labelling of all imported and exported data. Further, it requires restrictions on labels exported over each export channel. It also requires that labels be retained with the data exported. This facility is required at systems rated B1 or above.

2.3.10 Setting Communication Labels

2.3.10.1 Definition

The OSIF shall support features to set or change each communication channel and I/O device to either single-level or multi-level.

2.3.10.2 Metric

2.3.10.3 Rationale

The TCSEC (section 4.1.1.3.2) requires that communications channels and I/O devices (e.g., tape and disk drives) be marked as single-level or multi-level, and provide mechanisms to set or change that marking. This facility is required for systems rated B1 or above.

2.3.11 Identification and Authentication

2.3.11.1 Definition

The OSIF shall provide a protected mechanism to uniquely authenticate users' identity.

2.3.11.2 Metric

2.3.11.3 Rationale

The TCSEC (section 4.1.2.1) requires that a sign-on procedure be used to uniquely identify users. The technology required is not specified, although it is typically a user ID and password. Storage of authentication data must be protected (e.g., protection of passwords). This facility is required for all secure systems.

2.3.12 Labelling of Human Readable Output

2.3.12.1 Definition

The OSIF shall support the marking of human readable sensitivity labels on all human reAdable output.

2.3.12.2 Metric

2.3.12.3 Rationale

The TCSEC (section 4.1.1.3.2.3) requires labelling of all human reAdable output. This typically means labelling the top and bottom of each page of hard copy output. This facility is required for all systems at levels B1 and above.

2.3.13 Subject and Object Labelling

2.3.13.1 Definition

The OSIF shall support labelling (i.e., setting and changing) of each subject and object.

2.3.13.2 Metric

2.3.13.3 Rationale

The TCSEC (section 4.1.1.3) requires that each subject and object in the system be labelled. This facility is required of all systems at levels b1 and above.

2.3.14 Label Contents

2.3.14.1 Definition

The OSIF shall support the definition of a label for the system (i.e., classification, categories, markings, and special handling designators).

2.3.14.2 Metric

2.3.14.3 Rationale

The TCSEC (section 4.1.1.4) requires labels contain classifications and categories at the B1 level and above. Markings and special handling requirements are not explicitly required by TCSEC.

2.3.15 Mandatory Access Control Policy

2.3.15.1 Definition

The OSIF shall support a security policy based on subject and object labels.

2.3.15.2 Metric

2.3.15.3 Rationale

The TCSEC (section 4.1.1.4) requires that a policy exist and be enforced regarding access to subjects and objects based on a security policy. This requirement does not specify what the policy should be. However, TCSEC specifies characteristics of the policy (e.g., no write-down, no read-up, use of classifications and categories).

2.3.16 MAC Manipulations

2.3.16.1 Definition

The OSIF shall support the manipulation of labels based on the security policy.

2.3.16.2 Metric

2.3.16.3 Rationale

The TCSEC (section 4.1.1.4) requires that all manipulation of MAC labels be in accordance with the security policy for systems at level B1 or above.

2.3.17 Object Reuse

2.3.17.1 Definition

The OSIF shall provide that all objects are sanitized prior to allocation to a user.

2.3.17.2 Metric

2.3.17.3 Rationale

The TCSEC (section 4.1.1.2) requires such sanitization of all objects in the system to prevent unauthorized disclosure of information. Note that "objects" in this case refers to hardware elements (e.g., registers and memory) in addition to traditional objects such as files and disk space. This facility is required for systems at levels C2 or above.

2.3.18 User Notification of Sensitivity Label

2.3.18.1 Definition

The OSIF shall support the prompt notification to a terminal user of each change in security level associated with that user during an interactive session.

2.3.18.2 Metric

2.3.18.3 Rationale

The TCSEC (section 4.1.1.3.3) requires that the user be notified of such changes for levels B2 and above.

2.3.19 Sensitivity Label Query

2.3.19.1 Definition

The OSIF shall support the user's query of the subject's complete sensitivity label.

2.3.19.2 Metric

2.3.19.3 Rationale

The TCSEC (section 4.1.1.3.3) requires that the user have access to the complete sensitivity label (including special handling requirements) for systems at level B2 and above.

2.3.20 System Integrity

2.3.20.1 Definition

The OSIF shall support features that can be used to periodically validate the correct operation of the hardware and firmware.

2.3.20.2 Metric

2.3.20.3 Rationale

The TCSEC (section 4.1.3.1.2) requires that diagnostic and confidence tests be available to verify the correct functioning of the hardware at levels C1 and above.

2.3.21 Identification of Users Based on Roles

2.3.21.1 Definition

The OSIF shall support the identification of users based on roles.

2.3.21.2 Metric

2.3.21.3 Rationale

The TCSEC (section 4.1.1.4) requires that various roles be provided with varying privileges. at a minimum this separates the roles of operator, system administrator, and security administrator. This facility is required at levels B2 and above.

2.3.22 Least Privilege

2.3.22.1 Definition

The OSIF shall support the principle of least privilege.

2.3.22.2 Metric

2.3.22.3 Rationale

The TCSEC (section 4.1.3.1.4) requires that least privilege be used at levels B2 and above. The exact privileges required are not defined here.

2.3.23 Trusted Path

2.3.23.1 Definition

The OSIF shall support a trusted communication path between the user and the system, activated exclusively by the user.

2.3.23.2 Metric

2.3.23.3 Rationale

The TCSEC (section 4.1.2.1.1) requires that a trusted path be provided which provides for communication

between the system and the user without possibility of interception by any software other than the operating system. This facility is required at level B2 and above.

2.3.24 Trusted Recovery

2.3.24.1 Definition

The OSIF shall provide procedures and/or mechanisms to assure that after a discontinuity recovery without a protection compromise is obtained.

2.3.24.2 Metric

2.3.24.3 Rationale

The TCSEC (section 4.1.3.1.5) requires that systems at level B3 and above provide a trusted recovery mechanism. Such a mechanism is used for recovery after a failure (e.g., a system crash) which left the system in an unknown security state to a secure state.

2.4 DATA INTERCHANGE INTERFACES

2.4.1 Data Interchange Services (Data Format Conversion)

2.4.1.1 Definition

The OSIF shall support an access to services that perform data conversion, (e.g., files, CPUs, or compilers).

2.4.1.2 Metric

2.4.1.3 Rationale

Operating systems must handle data from various sources. In order to properly transmit data from one source to the other, and to perform operations on data, the operating system needs a canonical data format that is unaffected by the external environment. The operating system shall provide service routines that support conversion of one data format to another for a TBD set of processor internal data

2.5 EVENT AND ERROR INTERFACES

2.5.1 Event and Error Receipt

2.5.1.1 Definition

The OSIF shall provide for the receipt and coordination of event and error information.

2.5.1.2 Metric

2.5.1.3 Rationale

Event and error information includes information available through such interfaces as to the hardware/firmware, the network, and the program support environment as well as to the applications software. The Fault

Information collection requirement (Refer to Section 2.11.1) also applies to the error information collection part of this requirement.

2.5.2 Event and Error Distribution

2.5.2.1 Definition

The OSIF shall provide for the distribution of event and error information.

2.5.2.2 Metric

2.5.2.3 Rationale

This requirement is a necessary corollary to the Event and Error Receipt requirement (Refer to Section 2.5.1). The Fault Information Request requirement (Refer to Section 2.11.2) also applies to the error information distribution part of this requirement.

2.5.3 Event and Error Management

2.5.3.1 Definition

The OSIF shall support the timely delivery of interrupt and other asynchronous events to system components and shall support the implementation of user-selectable error processing alternatives. alternatives shall include as a minimum filtering, retry, ignore, and accumulate occurrences.

2.5.3.2 Metric

2.5.3.3 Rationale

For many applications the OSIF must provide for the timely delivery of interrupt and other signal information to system components such as the operator and application software. This includes, particularly in a tactical application, supporting the capability to display alerts to an operator at the system console. The Fault Detection Thresholds requirement (Refer to Section 2.11.6) also applies to this requirement.

2.5.4 Event Logging

2.5.4.1 Definition

The OSIF shall support logging events to application-defined storage. The types of events and event sources shall be dynamically selectable/deselectable.

2.5.4.2 Metric

2.5.4.3 Rationale

Examples of event sources are specific processors or memory modules. By providing event and source selectability, this requirement provides for saving information that is more relevant to the application while conserving storage resources.

2.5.5 Enable/Disable Interrupts

2.5.5.1 Definition

The OSIF shall provide the ability to enable and disable interrupts.

2.5.5.2 Metric

2.5.5.3 Rationale

This requirement provides for interrupts as a whole to be turned on and off. The mask/unmask interrupts requirement, on the other hand, provides for individual interrupts to be made known/unknown.

2.5.6 Mask/Unmask Interrupts

2.5.6.1 Definition

The OSIF shall provide the ability to mask and unmask events.

2.5.6.2 Metric

2.5.6.3 Rationale

A system requires this capability during such activities as interrupt processing to lock out interrupts of a lower class from occurring or to mask out the interrupts from particular

2.6 FILE INTERFACES

2.6.1 Contiguous Read of a File

2.6.1.1 Definition

The OSIF shall provide a capability to access information from an opened file in a contiguous stream.

2.6.1.2 Metric

2.6.1.3 Rationale

DBMS systems often read large blocks of data from the storage device to use as indices into other files. Such reads must be performed as quickly as possible. Limiting the number of seeks that a storage device must use (due to file fragmentation) can result in performance optimization. This function permits the access to a file that is contiguous on the storage medium.

2.6.2 Protect An Area Within A File

2.6.2.1 Definition

The OSIF shall provide a mechanism that restricts a file from access by requesters other than the requestor imposing the restriction.

2.6.2.2 Metric

2.6.2.3 Rationale

In order to protect simultaneous access and updates to a portion of files, the requestor of the access should be able to protect a section of the file until the user is finished updating. This is often called a file, byte, or record locking capability.

2.6.3 File Management Scheduling

2.6.3.1 Definition

The OSIF shall support a capability to specify a response requirement for the service being requested for file management.

2.6.3.2 Metric

2.6.3.3 Rationale

For hard deadline real-time systems file managers must schedule their service processing based on the response requirements of the requests submitted by the users. FIFO scheduling is unacceptable for real-time applications. The file managers must also support the notion of preemption.

2.6.4 File Management Suspend/Resume For Process

2.6.4.1 Definition

The OSIF shall permit a requesting process to indicate whether it wishes to wait for completion of the requested service before continuing processing or to continue without waiting. In support of the latter instance, a means shall be provided to enable the requesting task to know the status of its service request.

2.6.4.2 Metric

2.6.4.3 Rationale

In real-time applications it is often necessary to request data and then to suspend processing until the data is available. It is also often the case that the task has no need to wait on the completion of the service before it continues processing. For example, if the request is posting data to a file, it is usually unnecessary, even undesirable, for the task to wait, especially in a distributed environment.

2.6.5 File Management Block Requests

2.6.5.1 Definition

The OSIF shall support the capability to update or retrieve a set of contiguous records in a file.

2.6.5.2 Metric

2.6.5.3 Rationale

Often in real-time systems the position of a record in a file correlates to a time at which the information was derived. It is often necessary to read records that cover a particular span of time and to do so as expeditiously as possible. File managers should plan their accesses to minimize seek and latency when acting on these block requests.

2.6.6 Round Robin File Management

2.6.6.1 Definition

The OSIF shall support a form of file access wherein after a user specified number of records have been written, new records will begin to overlay old records. Also, support access requests for records that are based on relative position from newest/oldest record in file.

2.6.6.2 Metric

2.6.6.3 Rationale

Real-time systems keep historical files based on system requirements for coverage of some period of time. For example, two hours of track history may be required on-line for rapid access. If a record is written to the file every 10 seconds then the file will be 720 records long. The user
wants to update the file such that the 721st record overlays the 1st record. The user should not have to manage the file pointers.

2.6.7 Open a File

2.6.7.1 Definition

The OSIF shall provide a capability to open a file for use.

2.6.7.2 Metric

2.6.7.3 Rationale

The user of a file must be able to indicate that the file is in use so that the operating system can maintain the necessary status and buffers. This function can also become a form of protection for file usage. A possible implementation (or requirement) would be that the file would be available to read, write, execution, or deletion to one, a group, or many users.

2.6.8 Point Within a File

2.6.8.1 Definition

The OSIF shall provide a capability to position the next access point within a file.

2.6.8.2 Metric

2.6.8.3 Rationale

Some method must be provided to allow random access to a file on storage medium.

2.6.9 Read a File

2.6.9.1 Definition

The OSIF shall provide a capability to access information from an opened file.

2.6.9.2 Metric

2.6.9.3 Rationale

Some method must be provided to access data from a file on a storage medium.

2.6.10 Close a File

2.6.10.1 Definition

The OSIF shall provide a capability to close a file that has been opened.

2.6.10.2 Metric

2.6.10.3 Rationale

Operating systems often restrict the number of files that can be "open" at one time. In order to release files that are no longer needed, the operating system user should be able to close a file that is opened.

2.6.11 Delete a File

2.6.11.1 Definition

The OSIF shall provide a capability to have a file physically removed from a storage medium.

2.6.11.2 Metric

2.6.11.3 Rationale

In addition to maintaining an organized storage medium and minimizing information redundancy, the operating systems needs to provide a capability to delete files that are sensitive in nature.

2.6.12 Create a Directory

2.6.12.1 Definition

The OSIF shall provide a capability to add a directory to the storage structure on the storage medium.

2.6.12.2 Metric

2.6.12.3 Rationale

In order to have a useful file system, there must be some mechanism to maintain separate user areas on the file

storage medium. A possible implementation might be a hierarchical file system. Creating a directory enables an operating system user to add a user area.

2.6.13 Specifying Default Directory

2.6.13.1 Definition

The OSIF shall provide the capability to specify a processes default directory.

2.6.13.2 Metric

2.6.13.3 Rationale

In order to have a useful file system, there must be some mechanism to maintain separate user areas on the file storage medium. A possible implementation might be a hierarchical system. changing the current directory enables an operating system user to conveniently access a user area without providing path information.

2.6.14 Delete a Directory

2.6.14.1 Definition

The OSIF shall provide a capability to remove a directory from the storage structure on the storage medium.

2.6.14.2 Metric

2.6.14.3 Rationale

In order to have a useful file system, there must be some mechanism to maintain separate user areas on the file storage medium. A possible implementation might be a hierarchical system. Deleting a directory enables an operating system user to remove an unwanted user area.

2.6.15 Shadow Files

2.6.15.1 Definition

The OSIF shall support the creation, reading, writing, maintenance, and deletion of multiple identical instantiations of a file. The multiple instantiations shall be viewed at the OSIF boundary as a single object.

2.6.15.2 Metric

2.6.15.3 Rationale

Applications will need a mechanism to maintain files at multiple locations or redundantly to satisfy performance or reliability requirements.

2.6.16 Create a File

2.6.16.1 Definition

The OSIF shall provide a capability to create a file.

2.6.16.2 Metric

2.6.16.3 Rationale

The user must be able to create a file, declaring attributes to be associated with the file, to be used for subsequent application specific storage/retrieval.

2.6.17 Query File Attributes

2.6.17.1 Definition

The OSIF shall provide a capability to query the attributes of a file.

2.6.17.2 Metric

2.6.17.3 Rationale

Each file has attributes associated with it (e.g. file size, creation date, modification date, owner, permissions, storage type, access privileges). The user must be able to query the attributes in order to become aware of the current state of the file. This is especially true for attributes that are dynamic and for attributes that are not under direct control of the user.

2.6.18 Modify File Attributes

2.6.18.1 Definition

The OSIF shall provide a capability to modify the attributes of a file.

2.6.18.2 Metric

2.6.18.3 Rationale

Each file has attributes associated with it (e.g. file size, creation date, modification date, owner, permissions, storage type, access privileges). The user must be able to modify the attributes in order to respond to changing mission capabilities.

2.6.19 Write a File

2.6.19.1 Definition

The OSIF shall provide a capability to write information to an opened file.

2.6.19.2 Metric

2.6.19.3 Rationale

Some method must be provided to write data to a file on a storage medium.

2.6.20 Write Contiguous a File

2.6.20.1 Definition

The OSIF shall provide a capability to write information to an open contiguous file.

2.6.20.2 Metric

2.6.20.3 Rationale

Some method must be provided to write data to a contiguous file on a storage medium.

2.7 GENERALIZED I/O INTERFACES

In the following requirements device is used to indicate physical (i.e., a printer) or logical (i.e., pool of buffers) resources.

2.7.1 Device Driver Availability

2.7.1.1 Definition

The OSIF shall provide the interfaces necessary to support the addition of device drivers.

2.7.1.2 Metric

2.7.1.3 Rationale

In order for an operating system to be expandable, a new device driver will have to be added and an explicit interface is needed to do this without having to go back to the vendor to incorporate this new driver.

2.7.2 Open Device

2.7.2.1 Definition

The OSIF shall provide the ability for a process to request the services of a particular device.

2.7.2.2 Metric

2.7.2.3 Rationale

An interface is needed to allow processes to request devices in the system and use the services of that particular device. In Ada terms, this interface would be used by packages such as TEXT_IO when doing an OPEN on a file. In SAFENET terms, this interface may be used by applications wishing to use the primitive SA_REGISTER_req as outlined by the SAFENET standard. This primitive allows the user of SAFENET to receive a USAP (User Service Access Point) identification, which for all intents and purposes is a logical device that only that process or application can use, which gives it rights to the network.

2.7.3 Close Device

2.7.3.1 Definition

The OSIF shall provide the ability for a process to indicate that the services of a particular device, which had been previously allocated, are no longer needed.

2.7.3.2 Metric

2.7.3.3 Rationale

Once the device has been allocated and is no longer needed by a process, there needs to be a way to indicate to the operating system that this process is ready to release its control of the device. An example, in terms of Ada, of a process which would use this interface would be the TEXT_IO procedure CLOSE file. In SAFENET terms, this interface may be used by applications or processes wishing to use the primitive SA_CANCEL_req. This allows the application to give up its USAP identification, which is essentially saying its giving up its control of the logical device, which is access to the network.

2.7.4 Transmit Data

2.7.4.1 Definition

The OSIF shall provide the ability to transfer specified block(s) of data to a device which has been previously opened by a process.

2.7.4.2 Metric

2.7.4.3 Rationale

An interface to allow a process to communicate with a device which it has already acquired is needed so useful work can be done with this device. The TEXT_IO procedure PUT would be an example of a process which would use this interface. In SAFENET this interface would allow applications wishing to use the service which is indicated by the primitive SA_SEND_req access to that service.

2.7.5 Receive Data

2.7.5.1 Definition

The OSIF shall provide the ability to receive data from a device that has been previously opened by a process.

2.7.5.2 Metric

2.7.5.3 Rationale

Processes will need an interface to use which will allow them to receive data from a device which has been previously assigned to it. In terms of Ada, the package TEXT_IO would need this interface for its procedure named GET. The SAFENET service which is accessed by the primitive SA_REQUEST_req could be accessed by applications through this interface to receive data from other nodes in the system.

2.7.6 Device Event Notification

Refer requirements within Section 2.5: Event and Error Interfaces.

2.7.7 Control Device

2.7.7.1 Definition

The OSIF shall provide the mechanism to request a device to perform an action pertinent to the device.

2.7.7.2 Metric

2.7.7.3 Rationale

Processes need the capability to indicate some action to take place at a particular device through the operating system interface. An example, using Ada, may be the procedure found in the package TEXT_IO called RESET file. This procedure may need some interface to the operating system to carry out the action requested. The SAFENET service which is accessed by using the primitive SA_DISCONNECT_req could possibly be accessed by applications through this interface. an example would be sound an audible alarm, abort an ongoing activity on a device, and initialization of a device. Other examples may be sounding an alarm, aborting an ongoing activity on a device, etc.

2.7.8 I/O Directory Services

2.7.8.1 Definition

The Interface shall support the use of Directory Services to map between logical names and physical devices or address and attributes of the devices.

2.7.8.2 Metric

2.7.8.3 Rationale

A service must be provided to keep track of all the peripheral devices and their attributes. This service may be centralized or distributed. Its existence implies a name registration function and authority to insure the uniqueness of global names. The directory services function may be part of the basic operating system functions or it may be a part of the capabilities of a supporting subsystem such as SAFENET or a file management subsystem.

2.7.9 Device Management Suspend/Resume For Processes

2.7.9.1 Definition

The OSIF shall permit the requesting process to indicate whether it wishes to wait for completion of the requested service before continuing processing or to continue without waiting. In support of the latter instance, a means shall be provided to enable the requesting process to know the status of its service request.

2.7.9.2 Metric

2.7.9.3 Rationale

In real-time applications it is often necessary to request data and then to suspend processing until the data is available. It is also often the case that the process has no need to wait on the completion of the service before it continues processing. For example, if the request is posting data to a device, it is usually unnecessary, even undesirable, for the process to wait, especially in a distributed environment.

2.7.10 Mount/Dismount Device

2.7.10.1 Definition

The OSIF shall support the capability to mount and dismount a logical or physical device.

2.7.10.2 Metric

2.7.10.3 Rationale

Of particular concern are handling for traditional devices such as removable disk and tape storage entities. This mechanism may also be used to cause logical devices to become visible or invisible.

MOUNT - The action of mounting a logical or physical device causes that entity to become a visible resource that may be referenced by device identifier and commonly by some logical name associated with a particular instantiation of media associated with the device. Associated with the mount capability is the implied ability to specify access rules to be applied to the mounted entity.

2.7.11 Initialize/Purge Device

2.7.11.1 Definition

The OSIF shall support device-dependent initialization and deinitialization (purge) functions for logical and physical devices.

2.7.11.2 Metric

2.7.11.3 Rationale

Disk and tape devices (and media) need to be formatted, erased, labeled, unlabeled, etc. Logical devices can apply these same functions to provide functions such as network connection initialization, etc.

2.8 NETWORK AND COMMUNICATIONS INTERFACES

In a system using components based on NGCR standards there will frequently be a hierarchy of networked communication, data storage and processing functions. At the base of this hierarchy may be a number of processing or storage units on a single board connected by an on-board bus. At the next level will be FUTUREBUS+ or non-NGCR backplane busses (e.g. VME). At the next level there may be SAFENET, MIL-STD-1553B data busses or non-NGCR defined LANs. At the highest level, but outside the scope of this set of requirements, there may be communications among systems on different Navy platforms.

In some application domains and for some application functions, the OSIF must provide explicit access to networked communication, data storage, and processing functions for both NGCR-defined communication

components and similar non-NGCR-defined components. This is in addition to the implicit use of these capabilities implied in many other requirements.

Two processes make up a communications transaction regardless of their location. This includes either across a communications link or the two processes may possibly be residing on the same processor.

2.8.1 Interface to and Control of Navy Standard Inter Processing Unit Busses

2.8.1.1 Definition

The operating system shall provide explicit interfaces to and control of FUTUREBUS+, SAFENET, and MIL-STD-1553B in accordance with the standards or specifications defining each.

2.8.1.2 Metric

2.8.1.3 Rationale

These three sets of standards cover a broad range of capabilities. The OSIF, in addition to other functions, must provide the architecture, control and management structure to integrate these components into a usable and functioning whole.

FUTUREBUS+ is an emerging IEEE set of standards for backplanes used to interconnect processing units and other boards within a card cage. Among other capabilities it provides a precise common time of day clock and inter-process message passing for the FUTUREBUS+ interconnected devices.

SAFENET is the Navy's subset of International Standards Organization (ISO) Open System Interconnection (OSI) standards. These are supplemented by additional specifications and implementation agreements drawn from the Manufacturing Automation Protocol (MAP 3.0) specification, Government Open Systems Interconnection Profile (GOSIP, FIPSPUB 146) and SAFENET Working Group agreements. It provides services ranging from a variety of message communication services, to a file management and access system (FTAM), to support for the management of all components of the communication system. It permits users to also incorporate components based on ISO application layer standards not explicitly included in SAFENET.

MIL-STD-1553B is an older LAN with much lower performance characteristics than SAFENET. It only provides defined capabilities at the lower layers of the ISO/OSI model. However because of cost considerations and familiarity with its capabilities in the air community it may well continue to be used. The OS then must provide the architecture, control and management structure to integrate this component into the total system.

2.8.2 Interfaces to and Control of Other Network and Communication Entities

2.8.2.1 Definition

The OSIF shall support explicit interfaces to the capabilities of multiple standard and proprietary backplane busses and LANs.

2.8.2.2 Metric

2.8.2.3 Rationale

The OSIF must be capable of interfacing to a variety of proprietary and standard LANs and backplanes so as to support program transportability. This is needed to enable it to be used with specialized systems that cannot economically be modified to use Navy standards. In the area of backplanes VME and MULTIBUS are commonly used standards. Equipment such as DEC computers frequently use proprietary backplanes. In the area of LANs the INTERNET standards (particularly TCP/IP) are frequently used.

2.8.3 Reliable Virtual Circuit Communications

2.8.3.1 Definition

The OSIF shall provide for the selection of reliable virtual circuit communications.

2.8.3.2 Metric

2.8.3.3 Rationale

Certain applications require the ability to transfer data between processes via a connection oriented communications link. This link is established between processes and maintained for the transfer with error detection and correction support.

2.8.4 Unreliable Virtual Circuit Communications

2.8.4.1 Definition

The OSIF shall provide for the selection of unreliable virtual circuit communications.

2.8.4.2 Metric

2.8.4.3 Rationale

Certain applications call for the transfer of data over a connection oriented link between processes but can withstand a certain amount of error rather than the overhead associated with a reliable link. An example of such data is voice information.

2.8.5 Reliable Datagram Transfer

2.8.5.1 Definition

The OSIF shall provide for selection of reliable datagram transfer communications.

2.8.5.2 Metric

2.8.5.3 Rationale

Certain applications require the ability to transfer aperiodic information and do not require the establishment and maintenance associated with a connection oriented transfer. Yet they require an acknowledgment that the information was successfully received at the destination.

2.8.6 Unreliable Datagram Transfer

2.8.6.1 Definition

The OSIF shall provide for the selection of unreliable datagram transfer.

2.8.6.2 Metric

2.8.6.3 Rationale

Certain applications require the ability to transfer information without the overhead associated with connection oriented transfers and acknowledgments. These situations allow for the information to be sent with no assurance it properly arrives.

2.8.7 Request - Reply Service

2.8.7.1 Definition

The OSIF shall support the ability to select request - reply communication services.

2.8.7.2 Metric

2.8.7.3 Rationale

Certain applications require communication services in the form of a request and a reply. In these situations, a requesting process sends a datagram containing the request and associated data to a service process. Upon receipt of the request the service process performs any processing necessary and creates a datagram containing the request information and sends it to the requesting process.

2.8.8 Unreliable Broadcast/Multicast Service

2.8.8.1 Definition

The OSIF shall provide for the selection of an unreliable broadcast/multicast communication services.

2.8.8.2 Metric

2.8.8.3 Rationale

Certain applications require the ability to send a single message to all (broadcast) or several (multicast) destinations. In these situations it is sometimes desirable not to have the overhead associated with connection oriented transfers and reliable services.

2.8.9 Reliable Broadcast/Multicast Services

2.8.9.1 Definition

The OSIF shall provide for the selection of reliable broadcast/multicast communication services.

2.8.9.2 Metric

2.8.9.3 Rationale

Certain applications require the ability to send a single message to all (broadcast) or several (multicast) destinations. In these situations it is sometimes desirable to insure the proper reception of the information at all or some of the destinations.

2.8.10 Atomic Broadcast/Multicast Services

2.8.10.1 Definition

The OSIF shall provide for the selection of a reliable, atomic broadcast/multicast for communications services.

2.8.10.2 Metric

2.8.10.3 Rationale

Certain applications require the ability to send a single message to all (broadcast) or several (multicast) destinations. In these situations, synchronized behavior of the destinations is important so the communications subsystem must be able to guarantee that all destinations will receive the message within a stated time after the message is sent. by necessity, atomic messages must use a "reliable" broadcast/multicast service.

2.9 PROCESS MANAGEMENT INTERFACES

2.9.1 Create Process

2.9.1.1 Definition

The OSIF shall provide the ability to create processes with specified attributes.

2.9.1.2 Metric

2.9.1.3 Rationale

Processes and their environments need to be created prior to their execution. Attributes may include such things as process name, process priority, stack size, scheduling attributes, memory allocation, etc.

2.9.2 Terminate Process

2.9.2.1 Definition

The OSIF shall provide the ability to delete a process and recover all associated resources of that process.

2.9.2.2 Metric

2.9.2.3 Rationale

The OSIF must provide the service of deleting a process from being executed. In addition the operating system shall provide the ability of recovering all the deleted process resources if so directed.

2.9.3 Start Process

2.9.3.1 Definition

The OSIF shall provide a mechanism to designate a process as being ready to execute.

2.9.3.2 Metric

2.9.3.3 Rationale

The OSIF must proved the service of submitting a designated process to the processors scheduling queue.

2.9.4 Stop Process

2.9 4.1 Definition

The OSIF shall provide the ability to make a process unavailable for scheduling.

2.9.4.2 Metric

2.9.4.3 Rationale

There are situations where processes are stopped from execution yet remain in a "wait state" where they maybe restarted. Under these situations the OSIF must maintain the process and its environment, yet not consider it for scheduling until specifically notified.

2.9.5 Suspend Process

2.9.5.1 Definition

The OSIF shall provide the ability for a process to suspend itself or another process from execution such that the suspended process retains resources, rights and privileges and execution may be continued.

2.9.5.2 Metric

2.9.5.3 Rationale

The OSIF must provide the service of stopping a process from execution, yet maintain the process, the processes environment, and the processes data such that the process may be
"continued" from the point of execution at which it was suspended.

2.9.6 Resume Process

2.9.6.1 Definition

The OSIF shall provide the ability to continue the execution of a process that has been previously suspended.

2.9.6.2 Metric

2.9.6.3 Rationale

The OSIF shall provide a service to allow a previously suspended process to continue execution from the point at which it was suspended.

2.9.7 Delay Process

2.9.7.1 Definition

The OSIF shall provide the ability to delay the scheduling of a process for a specified time period.

2.9.7.2 Metric

2.9.7.3 Rationale

This service allows for a process to be identified for scheduling prior to the actual time it is desired for the process to be scheduled.

2.9.8 Interprocess Communication

2.9.8.1 Definition

The OSIF shall provide the ability for processes to exchange information.

2.9.8.2 Metric

2.9.8.3 Rationale

The OSIF must provide service(s) which allow processes to exchange data. These processes may or may not exist on the same processor. Examples of interprocess communication interfaces are shared files, lock files, shared memory, message passing, streams, pipes, fifos, signals, sockets, and access to higher level network services such as name servers and TCP/IP protocols.

2.9.9 Examine Process Attributes

2.9.9.1 Definition

The OSIF shall provide the ability for processes to examine the attributes of a particular process.

2.9.9.2 Metric

2.9.9.3 Rationale

Processes need the ability to read, analyze, and/or display the attributes of a particular process. Attributes may include such things as process name, process priority, stack size, scheduling attributes, memory allocation, etc.

2.9.10 Modify Process Attributes

2.9.10.1 Definition

The OSIF shall provide the ability for processes to modify the attributes of a particular process.

2.9.10.2 Metric

2.9.10.3 Rationale

Processes need the ability to modify the attributes assigned to a process when it was created when the significance of that process in the overall operation of the system changes. Examples of attributes that may require modification are process priority, stack size, scheduling attributes, memory allocation, etc.

2.9.11 Examine Process Status

2.9.11.1 Definition

The OSIF shall provide the ability for processes to examine the current status of a particular process.

2.9.11.2 Metric

2.9.11.3 Rationale

Processes need the ability to determine if another process has been created, started, deleted, stopped, suspended, etc.

2.9.12 Process Identification

2.9.12.1 Definition

The OSIF shall support the unambiguous identification of processes.

2.9.12.2 Metric

2.9.12.3 Rationale

Processes need the ability to identify other processes in the system in an unambiguous manner for such things as interprocess communication and examining the status of other processes. This includes different processes within the system and multiple copies of a single process within the system.

2.9.13 Save/Restart Process

2.9.13.1 Definition

The OSIF shall support the ability for processes to be restarted from a saved state.

2.9.13.2 Metric

2.9.13.3 Rationale

The state of a process (as reflected in its execution status and local environment) is often the cumulative result of hours of running within a mission critical system. It is commonly required in such systems to checkpoint the

state of critical processes so that they may be restarted from a known good state if hardware or software faults are later detected.

2.9.14 Program Management Function

2.9.14.1 Definition

The OSIF shall provide multiprogramming support.

2.9.14.2 Metric

2.9.14.3 Rationale

This permits multiple Ada "programs" to be active simultaneously within a common processor. This requires the assignment of memory and processing resources to the programs.

2.10 PROJECT SUPPORT ENVIRONMENT INTERFACES

2.10.1 Debug Support

2.10.1.1 Definition

The OSIF shall support the debugging of applications, specifically supporting the following capabilities:

- Examine Registers : the OSIF shall support a mechanism to examine registers of a selected resource in the system environment.
- Alter Registers : the OSIF shall support a mechanism to alter registers of a selected resource in the system environment.
- Set/Clear Breakpoint : the OSIF shall support a mechanism to set/clear multiple breakpoints.
- Set/Clear Watchpoints : the OSIF shall support a mechanism to set/clear multiple watchpoints.
- Single Step Execution : the OSIF shall support a mechanism to single step the execution of a software program.
- Continue Execution : the OSIF shall support a mechanism resume execution of a program after a breakpoint or watchpoint is encountered. The program shall resume execution at the next logical instruction.
- Examine Memory : the OSIF shall support a mechanism to read the contents of a process's address space.
- Alter Memory : the OSIF shall support a mechanism to modify the contents of a process' address space.
- Query Process Environment : the OSIF shall support a mechanism to examine the state of a process.

- Query Call Stack : the OSIF shall support the ability to determine the calling sequence of a process.

2.10.1.2 Metric

2.10.1.3 Rationale

Rationale for each of the ten required Debug capabilities is as follows:

- Examine Registers : In order to fully access the state of the system, the programmer must be able to access CPU registers.

- Alter Registers : In order to control the state of the machine at a given point in execution, the user must be able to modify register values.

- Set/Clear Breakpoint : debugging tools require the ability to halt execution of the code at pre-determined points to examine the state and status of the programming environment.

- Set/Clear Watchpoints : debugging tools require the ability to halt execution of the code when certain conditions or states occur to examine the state and status of the programming environment.

- Single Step Execution : debugging tools require the ability to examine the state and status changes that occur when each line of code (instruction) is executed.

- Continue Execution : debugging tools require the ability to resume normal execution of the program after it has been halted/stopped for examination of the programming environment.

- Examine Memory : debugging tools requires the ability to examine the memory within the address space of a program.

- Alter Memory : debugging tools requires the ability to modify the memory within the address space of a process.

- Query Process Environment : debugging tools require the ability to examine the state and status of the programming environment resulting from the execution of a process and does not exclude the states of associated queues and stacks (run, delay and entry queues, etc.).

- Query Call Stack : debugging tools require the ability to examine the trail of calling sequences (e.g., providing the ability to determine "How did we get here?").

2.10.2 Execution History

2.10.2.1 Definition

The OSIF shall support the ability to monitor the execution history of a process, including such information as:

- Frequency of calls

- Length of calls
- Missed deadlines
- Length of queues
- Tasking of runtime systems (e.g., number of context switches, CPU time used)
- Dynamic paging activity
- Memory allocation (e.g., number of requests, block sizes, fragmentation, length of use)
- What OS services are being used (e.g., passing labels)

2.10.2.2 Metric

2.10.2.3 Rationale

In order for a performance monitor to create a history of events, the OS must provide the above information.

2.11 RELIABILITY, ADAPTABILITY, MAINTAINABILITY INTERFACES

2.11.1 Fault Information Collection

2.11.1.1 Definition

The OSIF shall provide for specifying the collection of available fault information.

2.11.1.2 Metric

2.11.1.3 Rationale

An application must be able to determine that a non-recoverable fault has occurred, either by detecting the fault through information available to it or by receiving some signal from other systems/devices that a fault has been detected. This information is needed to increase the reliability of the system. This requirement provides a subset of the services required under Event and Error Interfaces (Refer to Section 2.5).

2.11.2 Fault Information Request

2.11.2.1 Definition

The OSIF shall provide for the receipt of fault information on request.

2.11.2.2 Metric

2.11.2.3 Rationale

The system must be able to determine that a non-recoverable fault has occurred, either by detecting the fault through information available to it or by receiving some signal from other systems/devices that a fault has been

detected. This information is needed to increase the reliability of the system. Receipt of fault information can be through active query or via a table that the application can access. This requirement provides a subset of the services required under Event and Error Management (Refer to Section 2.5.3).

2.11.3 Diagnostic Tests Reques

2.11.3.1 Definition

The OSIF shall provide for the initiation of diagnostic tests on specific request. The OSIF shall support initiation of diagnostic tests at specified intervals.

2.11.3.2 Metric

2.11.3.3 Rationale

Examples of these tests are Built In Test Equipment (BITE) tests, when software can initiate them, and firmware diagnostic tests of hardware components.

2.11.4 Diagnostic Tests Results

2.11.4.1 Definition

The OSIF shall provide the ability to determine the results of diagnostic tests.

2.11.4.2 Metric

2.11.4.3 Rationale

Receipt of the results of diagnostic tests can be through active query or via a table that the application can access. These diagnostic tests can be those initiated under the Diagnostic Tests Request requirement (Refer to Section 2.11.3) or self tests independently initiated by the affected system component.

2.11.5 Operational Status

2.11.5.1 Definition

The OSIF shall provide access to the operational status of all system components.

2.11.5.2 Metric

2.11.5.3 Rationale

System components include both software and hardware components such as buses, memory modules, processors, and I/O channels. Status indications include on, off, faulty, suspect, and the relief of a previously reported fault or overload condition.

2.11.6 Fault Detection Thresholds

2.11.6.1 Definition

The OSIF shall provide for specifying fault detection thresholds. These shall include but not be limited to:

- number of retry attempts, if applicable, that shall be made before an error is determined to be a non-recoverable fault
- maximum number of correctable errors that, if detected within a specified time, will classify the component as suspect or treat the collective errors as a non-recoverable fault

2.11.6.2 Metric

2.11.6.3 Rationale

The thresholds cited in the definition are required to detect intermittent faults. This requirement also applies to the Event and Error Management requirement (Refer to Section 2.5.3).

2.11.7 Fault Isolation

2.11.7.1 Definition

The OSIF shall support the isolation of faults to a particular component.

2.11.7.2 Metric

2.11.7.3 Rationale

Not only must an operating system interface provide detailed error information but it must also support localizing the fault so that applications software can be reconfigured and equipment repaired or replaced. Component, as used in the definition, refers to both hardware and software components.

2.11.8 Fault Response

2.11.8.1 Definition

The OSIF shall provide for the specification of actions to be taken on the occurrence of a fault. The OSIF shall support at least the following actions:

- Restart at a specified point for a specified fault
- Use of specified components as backup for faulty components
- Stop when a specified minimum set of components is no longer available
- Schedule a specified process
- Report to another node

2.11.8.2 Metric

2.11.8.3 Rationale

Navy applications, particularly those that are platform deployed, have traditionally required ever increasing fault tolerant coverage. Part of that coverage has included providing a variety of fault responses to cover not only various kinds of faults but also various mission and processing requirements.

2.11.9 Reconfiguration

2.11.9.1 Definition

The OSIF shall support the dynamic reconfiguration of hardware and software.

2.11.9.2 Metric

2.11.9.3 Rationale

The set of available configurations for a particular implementation can be predetermined at system build time. These configurations will specify various configurations of the software to accommodate such variables as changes in mission requirements and operating in degraded modes. They will also specify the configurations that make sense for an implementation such as minimum memory requirements. The purpose of this requirement is to allow an implementation to make the best use of available hardware and software resources.

2.11.10 Enable/Disable System Component

2.11.10.1 Definition

The OSIF shall provide the ability to enable or disable a specified system component on request.

2.11.10.2 Metric

2.11.10.3 Rationale

This requirement supports reconfiguration. Examples of hardware components are processors, memory modules, and buses. Groups of software components could be subsystems or Ada programs.

2.11.11 Performance Monitoring

2.11.11.1 Definition

The OSIF shall support queries for snapshots of resource utilization. The OSIF shall support enabling or disabling monitoring of each resource.

2.11.11.2 Metric

2.11.11.3 Rationale

Snapshots are defined to be of a specified time exposure (as opposed to instantaneous).

2.11.12 Set Resource Utilization Limits

2.11.12.1 Definition

The OSIF shall support pre-defining and dynamically adjusting over and under utilization limits for a process on a specified resource.

2.11.12.2 Metric

2.11.12.3 Rationale

Limits can be set at system build and then modified runtime per process and resource combination.

2.11.13 Resource Utilization Limits Violation

2.11.13.1 Definition

The OSIF shall support the detection and reporting of a process which violates its utilization limits for a resource.

2.11.13.2 Metric

2.11.13.3 Rationale

Once a limit is violated the application may examine overall system performance and the situation to determine if a fault exists or if this load is consistent with operating demands. This requirement correlates directly to the Set Resource Utilization Limits requirement, 2.11.12.

2.11.14 Checkpoint Data Structures

2.11.14.1 Definition

The OSIF shall support the ability to replace specified existing data structures with those same structures as they appeared at a certain point in the past.

2.11.14.2 Metric

2.11.14.3 Rationale

Reconfiguration and diagnostics require the ability to move data structures in and out of memory. For example, in the event of the failure of a global memory module and consequent reconfiguration, applications could be warm-started in place, passing a pointer to the last checkpointed copies of mission-critical data structures.

2.12 RESOURCE MANAGEMENT INTERFACES

2.12.1 Virtual Memory Support

2.12.1.1 Definition

The OSIF shall support the selection of the virtual memory utilization parameters.

2.12.1.2 Metric

2.12.1.3 Rationale

On processor architectures supporting a larger virtual address space than the size of physical memory, the operating system implementation will generally support the virtual memory mapping hardware. The paging algorithm used, and other virtual memory support parameters will need to be tailored to the application.

2.12.2 Virtual Space Locking

2.12.2.1 Definition

The OSIF shall provide the capability to lock certain application specified regions of virtual code and data space into physical memory, and for the subsequent release of such locks.

2.12.2.2 Metric

2.12.2.3 Rationale

For time-critical portions of applications, paging data and/or code to a secondary (mass) storage device would not allow for high performance access. For fault tolerant applications, the fault handling logic cannot be placed on a device that is likely to fail.

2.12.3 Dynamic Memory Allocation and Deallocation

2.12.3.1 Definition

The OSIF shall provide for allocation of a block of virtual or physical memory of the size specified and for deallocation of a previously allocated block.

2.12.3.2 Metric

2.12.3.3 Rationale

An application entity may require a global heap for its own dynamic memory management (e.g. the Ada run-time library), for dynamic load or relocation of code, for temporary buffers, etc. Such blocks, when no longer required by the application, should be re-entered into the pool of available physical memory.

2.12.4 Dynamic Memory Protection

2.12.4.1 Definition

The OSIF shall provide the ability to query and set memory protection attributes.

2.12.4.2 Metric

2.12.4.3 Rationale

Mission critical systems must guard against erroneous memory references (whether the result of software bugs, a security breach, or a hardware fault). While there is no foolproof approach to this, hardware memory protection provides a substantial level of confidence; but only if the OS interface provides for tailoring the

memory protection to the application's needs. Any arbitrary block of memory may contain code, read/write data, read-only data, or (perhaps in multi-level secure systems) write-only data. Memory protection requirements on a block may change over its lifetime.

Specification is required for all static code and data areas; any block of memory obtained through dynamic memory allocation may have its attributes specified during allocation. Memory protection attributes for any (static or dynamic) block should be alterable at run-time. Protection violations should result in error events (Refer to Section 2.5.3).

2.12.5 Shared Memory

2.12.5.1 Definition

The OSIF shall support concurrent access, by several processes, to specified areas of physical memory.

2.12.5.2 Metric

2.12.5.3 Rationale

The concept of Ada library units requires shared memory for both code and data. Time critical applications often cannot tolerate the overhead of message passing, rendezvous, or other forms of interprocess (or inter-task) communication. Applications are responsible for sensible use of the shared memory resource (see Synchronization and Scheduling Interface, Mutual Exclusion).

For virtual storage architectures, this will require a many-to-one mapping from virtual memory spaces to the shared physical page(s). Where the several processes are distributed across several processors separated by backplane or network interfaces, this will implicitly require interprocessor communication and synchronization.

2.12.6 Allocate, Deallocate, Mount, Dismount Services

2.12.6.1 Definition

The OSIF shall support the allocation of devices to processes, and subsequent deallocation of these devices. For devices with removable media, the OSIF shall also support mounting and dismounting of media.

2.12.6.2 Metric

2.12.6.3 Rationale

It is in the nature of some devices that they may be opened by several processes (i.e. shared), but many devices must be accessed exclusively by one process at a time. Some devices support opening of mountable volumes, and the OS should also provide explicit interfaces to specify the mounting and dismounting of such volumes. Control over such details is often left to an ad-hoc interface to device drivers, but these common requirements are better handled via explicit application/OS/device-driver interfaces.

2.12.7 Designate Control

2.12.7.1 Definition

The OSIF shall provide the means to designate responsibility for maintaining the status and determining the configuration of a system resource.

2.12.7.2 Metric

2.12.7.3 Rationale

A basic purpose of an operating system is to regulate the control of system resources. This interface may be pre-run-time (static designation of control) or run-time (dynamic designation of control). The unit of software assuming the responsibility may be the operating system itself.

2.12.8 Release Control

2.12.8.1 Definition

The OSIF shall provide the means to release a previously assumed system resource status and configuration responsibility.

2.12.8.2 Metric

2.12.8.3 Rationale

Responsibilities that software can assume at runtime need also to be able to be revoked and reassigned. This shall allow the operating system to designate responsibility for the system resource to another unit of software via the "Designate Control" interface.

2.12.9 Allocate Resource

2.12.9.1 Definition

The OSIF shall provide a means to designate particular process resources for use by a particular process.

2.12.9.2 Metric

2.12.9.3 Rationale

A basic purpose of an operating system is to regulate the control of system resources. The allocation request shall actually be honored by the entity currently designated as controlling the resource. Examples of units of system resources are an I/O channel, a block of physical memory, response to a specific class of hardware interrupt, a breakpoint register, a co-processor user identifier, and a connection over a LAN. The software making the allocation may be the operating system itself or may be application software assuming status and configuration responsibilities.

2.12.10 Deallocate Resource

2.12.10.1 Definition

The OSIF shall provide a means to relinquish particular system resources from a particular process.

2.12.10.2 Metric

2.12.10.3 Rationale

Resources that software can assume at runtime need also to be able to be revoked and reassigned.

2.12.11 System Resource Requirements Specification

2.12.11.1 Definition

The OSIF shall provide the ability to specify system resource requirements.

2.12.11.2 Metric

2.12.11.3 Rationale

The ability to modify the allocation of system resources based on operational need is supported by this requirement. Specification of resource requirements before requesting resource allocation is required for effective management of resources, especially to prevent deadlock among contenders for the resources.

2.12.12 System Resource Capacity

2.12.12.1 Definition

The OSIF shall provide a query of the storage or workload capacities of the system resources.

2.12.12.2 Metric

2.12.12.3 Rationale

The application (or entity controlling a resource) needs to know the availability and capacity of a resource to effectively allocate it during system operation.

2.13 SYNCHRONIZATION AND SCHEDULING INTERFACES

2.13.1 Process Synchronization

2.13.1.1 Definition

The OSIF shall provide an explicit mechanism by which two processes may synchronize their execution.

2.13.1.2 Metric

2.13.1.3 Rationale

Processes require the ability to synchronize their execution in real time applications.

In order to ensure predictable performance this may include access to low-level synchronization mechanisms to ensure proper communication protocols.

Synchronization should prohibit priority inversion situations.

2.13.2 Mutual Exclusion

2.13.2.1 Definition

The OSIF shall provide mutual exclusion and shall support mutual exclusion with timeouts.

2.13.2.2 Metric

2.13.2.3 Rationale

The system must have a low level mutual exclusion mechanism available to all users. If mutual exclusion is implemented with semaphores, then the semaphores must have operations available to create/destroy them, to claim/release them, and for priority queuing of processes waiting for a semaphore. Time-out mechanisms for processes waiting on semaphores must also be available. Developers must also be able to query the status of a semaphore. All resources must have the ability to control critical sections for mutual exclusion. This is necessary for both safety and security. Processes that request a shared resource must have the ability/option to withdraw their request via a time-out, which may be zero; i.e., immediate withdrawal if the resource is not immediately available.

2.13.3 Cumulative Execution Time of a Process

2.13.3.1 Definition

The OSIF shall provide the ability to access the cumulative execution time of a process.

2.13.3.2 Metric

2.13.3.3 Rationale

When the scheduler is responsible for aperiodic process, it needs a means of determining the cumulative execution time so that priorities of the processes may be adjusted. The operating system interface must provide a means for applications software to monitor and establish the rules for scheduling and execution of aperiodic processes (Refer to Section 2.13.10).

2.13.4 Attach a Process to an Event

2.13.4.1 Definition

The OSIF shall support the ability to attach a process to an event.

2.13.4.2 Metric

2.13.4.3 Rationale

The application must be able to provide the scheduler the information necessary to attach a process to an interrupt. This allows a process to respond to an external stimulus and helps to obtain more flexible scheduling. An example of an attached process is an event handler.

2.13.5 Transaction Scheduling Information

2.13.5.1 Definition

The OSIF shall provide the ability for a process to specify its response requirements for services.

2.13.5.2 Metric

2.13.5.3 Rationale

Scheduling in hard real time systems must be done in a fashion to meet deadlines. For transactions that require a sequence of operations across a distributed system, the scheduling mechanisms involved require the ability of scheduling processes with respect to deadline requirements.

2.13.6 Scheduling Delay

2.13.6.1 Definition

The OSIF shall support the ability to delay the scheduling of a process.

2.13.6.2 Metric

2.13.6.3 Rationale

(Refer to Section 2.9.7)

2.13.7 Periodic Scheduling

2.13.7.1 Definition

The OSIF shall provide for the periodic scheduling of a process.

2.13.7.2 Metric

2.13.7.3 Rationale

In real time systems, certain process require the ability to be scheduled at a specific periodic rate. The rate may be specified with respect to a mean delta with a plus and minus limit of variance.

2.13.8 Multiple Scheduling Policies

2.13.8.1 Definition

The OSIF shall support multiple scheduling policies.

2.13.8.2 Metric

2.13.8.3 Rationale

Different applications require that different scheduling algorithms.

2.13.9 Selection of a Scheduling Policy

2.13.9.1 Definition

The OSIF shall support the ability to select the scheduling policy to suit the need.

2.13.9.2 Metric

2.13.9.3 Rationale

It is perceived that once a scheduling algorithm is selected for an application, it remains static under normal conditions. However, mode changes or workload extremes may require dynamic alterations in scheduling policies. To meet this requirement, scheduling policies must be able to be altered without system reinitialization.

2.13.10 Modification of Scheduling Parameters

2.13.10.1 Definition

The OSIF shall support the ability to modify the values of the controllable scheduling parameters.

2.13.10.2 Metric

2.13.10.3 Rationale

Certain applications require the ability to dynamically modify the scheduling algorithms parameters used for selection of the process to be submitted for execution.

The scheduler will need the freedom to change the priority of a process dynamically. As the system operates, different processes will assume prominent positions and therefore require higher priorities. This adjustment of priorities must be dynamic in order to maximize system performance. The policies by which the priority adjustments are made must be controlled by the application software.

2.13.11 Precise Scheduling (Jitter Management)

2.13.11.1 Definition

The OSIF shall provide the ability for an application to indicate to the scheduler an exact specified time for starting a process.

2.13.11.2 Metric

2.13.11.3 Rationale

The scheduler must be able to guarantee the process is executed at the exact time specified and is not unduly delayed.

In real-time systems, completion of a scheduling event too early can be as bad as completion of a scheduling

event too late, i.e., to miss a deadline. This phenomena is known as "jitter" and can cause performance problems in real-time systems. In order for real-time systems to perform as predicted, and to ensure stability, schedule must be met as closely as possible. It is not appropriate to complete an event early if it can be avoided. This is one of the things that separates real-time systems from time-sharing systems.

2.14 SYSTEM INITIALIZATION AND REINITIALIZATION INTERFACES

2.14.1 Image Load

2.14.1.1 Definition

The OSIF shall provide the capability to perform initial and reinitial executable image load (including data) both locally and remotely to and for each and all processor(s) throughout a system.

2.14.1.2 Metric

2.14.1.3 Rationale

The OSIF must support and provide the capability to load and reload initialize and reinitialize an executable image into each and all processor(s) throughout a system, both locally and remotely. This includes initial (cold start) and reinitial (cold, reconfigured (re)start and/or warm, reconfigured (re)start) of the OSs own designated processor and all others.

2.14.2 System Initialization and Reinitialization

2.14.2.1 Definition

The OSIF shall support the capability to initialize and reinitialize all system resources.

2.14.2.2 Metric

2.14.2.3 Rationale

A distributed, multiple processor, real-time system must be initialized from a cold start and reinitialized after a cold start or warm (re)start such that the system configuration information necessary to execute the functions of the system is properly loaded in the different system components. This includes all communications, input/output ports, data storage and access components, etc. This means that the Operating System Interface must support all necessary system initialization and reinitialization functions for a given application. This is not limited to image load, initialization, or re-initialization.

2.14.3 Shutdown

2.14.3.1 Definition

The OSIF shall provide the capability to perform planned, orderly shutdown at the local and remote levels for each and all processor(s) throughout a system.

2.14.3.2 Metric

2.14.3.3 Rationale

The OSIF must provide the capability to perform planned, orderly shutdown operations when required under crisis and non-crisis situations. This is good resource management policy in all situations up to the most catastrophic crash event in order to attempt an orderly recovery.

2.15 TIME SERVICES INTERFACES

2.15.1 Read Selected Clock

2.15.1.1 Definition

The OSIF shall provide the ability to read selected clocks.

2.15.1.2 Metric

2.15.1.3 Rationale

The OSIF must have a facility for applications to read a selected clock, or set of clocks, in a system. Many applications have the need to time stamp data either to coordinate events taking place in different parts of the system or to record when events take place so that data may be later properly processed or time ordered.

2.15.2 Set Selected Clock

2.15.2.1 Definition

The OSIF shall provide the ability to set selected clocks.

2.15.2.2 Metric

2.15.2.3 Rationale

The clocks used in a system may change over time. They may need to be set when a system component is initialized. The clock is a resource whose detailed management belongs to the OS. However the setting of it and coordination with external time sources are issues which must be left to the designer of a specific system. The OSIF as the controller of common resources must have a facility for applications to set a selected clock, or set of clocks, in a system.

2.15.3 Synchronization of Selected Clocks

2.15.3.1 Definition

The OSIF shall support the ability to selectively synchronize clock(s) in the system.

2.15.3.2 Metric

2.15.3.3 Rationale

The OSIF must have a facility for applications to selectively synchronize clocks in a system. The facility must

allow synchronization of one clock or set of clocks to other clock sets in the system. These clock sets may be part of different subsystems, for example SAFENET, FUTUREBUS+, and a navigation system clock synchronized to Greenwich Mean Time. A means must exist to synchronize these to support those cases in which a common platform time base is required.

2.15.4 Select a Primary Reference Clock

2.15.4.1 Definition

The OSIF shall support the ability to select a primary reference clock for the system.

2.15.4.2 Metric

2.15.4.3 Rationale

The OSIF must have a facility for applications to select one primary reference clock or a set of primary clocks out of all clocks in a system and the set of systems integrated on a platform. This primary set must be able to be used to support clock synchronization throughout the system and also support the selection of a backup reference clock in the event of failure of the primary. This is required to support those cases in which clocks of different quality are used in different subsystems or within a subsystem. A method must exist for indicating that a high quality clock or set of clocks shall be used as the reference set to which others are synchronized.

2.15.5 Locate the Primary Reference Clock

2.15.5.1 Definition

The OSIF shall support the ability to locate the primary reference clock for a system.

2.15.5.2 Metric

2.15.5.3 Rationale

The OSIF must have a facility for applications to locate the primary reference clock in a system. This is required to support system management functions. For example, a failure in a system component could cause a system manager application to lose track of the identity of the current primary clock.

2.15.6 Timer Services

2.15.6.1 Definition

The OSIF shall support the setting and clearing of alarms and shall allow for notification at alarm time. The alarm time would be inclusive of either relative time difference or absolute time difference.

2.15.6.2 Metric

2.15.6.3 Rationale

The OSIF must have a facility for applications to set alarms for such things as watchdog timers, delays, etc. Once the time has expired the notification of the alarm must be propagated to the appropriate recipient of the

alarm.

An Ada application and the Ada runtime system must be able to specify delays in terms of either an absolute or relative time basis. examples:

```
delay 2.0;                          -- delay for 2 seconds from now.
delay_until(Next_Time); -- delay until the absolute time
                                    -- specified by the variable
                                    -- Next_Time
```

2.15.7 Precision Clock

2.15.7.1 Definition

The OSIF shall provide a time granularity of one nanosecond to processes that is independent of the granularity of the underlying hardware.

2.15.7.2 Metric

2.15.7.3 Rationale

Applications must have a consistent, portable interface to the underlying clock(s) that allow them to use the full capability of the clock(s).

2.16 ADA LANGUAGE SUPPORT INTERFACES

2.16.1 Create Task (Ada)

2.16.1.1 Definition

The OSIF shall support the capability to create an Ada task that supports the full set of Ada tasking operations as defined in the Ada Language Reference Manual (ANSI/MIL-STD-1815A).

2.16.1.2 Metric

2.16.1.3 Rationale

An Ada runtime system must have the ability to create Ada tasks as logically concurrent threads of execution that are managed by the Operating System. At the point of task creation, it must be possible to specify the Ada task's attributes (e.g., task name, priority, the task's master, stack space size, the number of entries) and the system resources (e.g., memory) needed to support the execution of the created Ada task (Refer to Section 2.9.1).

2.16.2 Abort Task (Ada)

2.16.2.1 Definition

The OSIF shall support the capability to abort the execution of an Ada task as defined in the Ada Language Reference Manual

(ANSI/MIL-STD-1815A).

2.16.2.2 Metric

2.16.2.3 Rationale

An Ada runtime system must have the ability to abort Ada tasks and recover the resources previously held by those tasks (Refer to Sections 2.9.2 and 2.9.4).

2.16.3 Suspend Task (Ada)

2.16.3.1 Definition

The OSIF shall support the capability to suspend the execution of an Ada task.

2.16.3.2 Metric

2.16.3.3 Rationale

An Ada runtime system must have the ability to suspend the execution of Ada tasks in order to support various task scheduling mechanisms (Refer to Section 2.9.5).

2.16.4 Resume Task (Ada)

2.16.4.1 Definition

The OSIF shall support the capability to resume the execution of an Ada task.

2.16.4.2 Metric

2.16.4.3 Rationale

An Ada runtime system must have the ability to resume the execution of Ada tasks in order to support various task scheduling mechanisms (Refer to Section 2.9.6).

2.16.5 Terminate Task (Ada)

2.16.5.1 Definition

The OSIF shall support the capability to terminate the execution of an Ada task as defined in the Ada Language Reference Manual (ANSI/MIL-STD-1815A).

2.16.5.2 Metric

2.16.5.3 Rationale

An Ada runtime system must have the ability to terminate the execution of Ada tasks in order to support the full semantics of the Ada tasking Model.

2.16.6 Restart Task (Ada)

2.16.6.1 Definition

The OSIF shall support the capability to restart the execution of an Ada task at a point immediately following its elaboration code.

2.16.6.2 Metric

2.16.6.3 Rationale

An Ada runtime system must have the ability to restart the execution of Ada tasks in order to support mode change operations (Refer to Section 2.9.13).

2.16.7 Ada Task Entry Calls

2.16.7.1 Definition

The OSIF shall support simple, timed, and conditional Ada task entry calls as defined in the Ada Language Reference Manual (ANSI/MIL-STD-1815A).

2.16.7.2 Metric

2.16.7.3 Rationale

An Ada runtime system must have the ability to implement all forms of Ada task entry calls, namely simple, timed, and conditional calls.

2.16.8 Ada Task Call Accepting/Selecting

2.16.8.1 Definition

The OSIF shall support the various forms of accepting Ada task entry calls as defined in the Ada Language Reference Manual (ANSI/MIL-STD-1815A). In particular, the OSIF shall support simple accepts, simple selective waits, selective waits with delay alternatives, selective waits with an else clause, and selective waits with a terminate alternative.

2.16.8.2 Metric

2.16.8.3 Rationale

An Ada runtime system must have the ability to implement all forms of accepting and selecting Ada task entry calls as defined in the Ada Language Reference Manual (ANSI/MIL-STD-1815A).

2.16.9 Access Task Characteristics (Ada)

2.16.9.1 Definition

The OSIF shall support the capability to access an Ada task's attributes and characteristics.

2.16.9.2 Metric

2.16.9.3 Rationale

An Ada runtime system must have the ability to read a task's attributes (e.g., task ID, execution state, available CPU time compared with specified time budget), and also, to read and write a task's characteristics (e.g., priority, period, phase) in order to implement various scheduling mechanisms (Refer to Section 2.9.10).

2.16.10 Monitor Task's Execution Status (Ada)

2.16.10.1 Definition

The OSIF shall support the ability to monitor a task's execution status, in particular, the amount of accumulated CPU time that has been used by the task.

2.16.10.2 Metric

2.16.10.3 Rationale

An Ada runtime system must have the ability to monitor a task's execution behavior in terms of the amount of accumulated CPU time that it has used. Such information is needed on a task-by-task basis in order to implement certain real-time scheduling algorithms (e.g., deferrable server, sporadic server, degraded mode for imprecise results) (Refer to Section 2.9.11).

2.16.11 Access to a Precise Real-Time Clock (Ada)

2.16.11.1 Definition

The OSIF shall support access (e.g., read/write, setting alarms) to a precise, continuous real-time clock.

2.16.11.2 Metric

2.16.11.3 Rationale

An Ada runtime system must have the ability to read from and write to a precise real-time clock. Also, the Ada runtime must be able to set or remove timer alarms to be triggered at a specified time in the future, in order to implement Ada's delay statement and timed entry calls. Setting these timer alarms is also useful for implementing (1) precise, periodic scheduling of Ada tasks, (2) watchdog timers, and (3) timeouts on communication primitives (Refer to Sections 2.15.6 and 2.15.7).

2.16.12 Access to a Time-of-Day Clock (Ada)

2.16.12.1 Definition

The OSIF shall support read and write access to a time-of-day (TOD) clock.

2.16.12.2 Metric

2.16.12.3 Rationale

An Ada runtime system must have the ability to read from and write to a time-of-day clock in order to support the operations defined in package calendar. Furthermore, an Ada application program must be able to read the TOD (through a calendar.Clock function call) clock in order to affect a delay until a specified time in the future (e.g., delay (Next_Start_Time - calendar.Clock)).

2.16.13 Dynamic Task Priorities (Ada)

2.16.13.1 Definition

The OSIF shall support the capability to get and set the execution priority of an Ada task.

2.16.13.2 Metric

2.16.13.3 Rationale

An Ada runtime system must have the ability to dynamically control the execution priority of an Ada task in order to implement various scheduling mechanisms (Refer to Section 2.13.10).

2.16.14 Scheduling Policy Selection (Ada)

2.16.14.1 Definition

The OSIF shall support the capability to get and set the policy that is to be used to schedule Ada tasks.

2.16.14.2 Metric

2.16.14.3 Rationale

An application must be able to select the scheduling policy (e.g., priority preemptive, time slicing within equal priority levels) that will be used by the Operating System to schedule executing tasks. Open issues: (1) one versus multiple policies in effect at a given time; (2) how various policies interact, and (3) the scope of a scheduling policy (e.g., entry queues, run queue).

2.16.15 Memory Allocation and Deallocation (Ada)

2.16.15.1 Definition

The OSIF shall support the capability to create and/or delete a pool of memory that can be used as a heap for allocation and deallocation of smaller access collections. Furthermore, the OSIF shall support the capability to allocate data objects from both an independently allocated heap (e.g., Ada access collection) and a global pool of unallocated memory. It must be possible for the application to notify the Operating System when use of the heap space is no longer required.

2.16.15.2 Metric

2.16.15.3 Rationale

An Ada runtime system must have the ability to allocate memory space for access variables (i.e., access collections) and task stacks. Heap management is necessary in order to prevent memory fragmentation and

other garbage collection related problems, and to allocate and deallocate large chunks of memory based on dynamic scope (Refer to Section 2.12.3).

2.16.16 Interrupt Binding (Ada)

2.16.16.1 Definition

The OSIF shall support the capability to bind and unbind an interrupt to Ada application code, in particular, to at least an Ada interrupt task entry.

2.16.16.2 Metric

2.16.16.3 Rationale

An Ada runtime system must have the ability to attach and detach code to a device interrupt. The Ada Language Reference Manual (LRM) suggests that interrupts can be bound to task entries using an address clause. Morcover, a conventional Interrupt Service Routine (ISR) approach requires that the ISR code be directly tied to a device interrupt (Refer to Section 2.13.4).

2.16.17 Enable/Disable Interrupts (Ada)

2.16.17.1 Definition

The OSIF shall support the capability to enable and disable interrupts.

2.16.17.2 Metric

2.16.17.3 Rationale

Ada applications and the Ada runtime system must have the ability to control interrupts by enabling and disabling them. Often times, controlling interrupts is used as a programming technique for implementing critical sections of code. Disabling and enabling interrupts is also necessary for controlling a device's operations.

2.16.18 Mask/Unmask Interrupts (Ada)

2.16.18.1 Definition

The OSIF shall support the capability to mask and unmask device interrupts.

2.16.18.2 Metric

2.16.18.3 Rationale

Ada applications and the Ada runtime system must have the ability to control device interrupts by masking and unmasking them. Masking and unmasking interrupts is also necessary for controlling a device's operations.

2.16.19 Raise Exception (Ada)

2.16.19.1 Definition

The OSIF shall support the capability to raise an exception in an Ada task.

2.16.19.2 Metric

2.16.19.3 Rationale

An Ada runtime system must have the ability to raise an exception in any given Ada task. In particular, an Ada runtime system must be able to raise an exception in a task when hardware-detected exceptions (e.g., overflow, access violation) occur.

2.16.20 Ada Input/Output Support

2.16.20.1 Definition

The OSIF shall support for Ada input/output as described in Chapter 14 of the Ada Language Reference Manual.

2.16.20.2 Metric

2.16.20.3 Rationale

Conformance to the Ada Language Standard. The correspondence between the input/output supported for Ada and all other input/output supported by the interface must be clearly defined. The interface must provide access from Ada to files written by other languages (if any).

•

NAVSWC TR 90-248

EVALUATION PROCESS REPORT FOR NEXT GENERATION COMPUTER RESOURCES OPERATING SYSTEMS INTERFACE BASELINE SELECTION

BY NEXT GENERATION COMPUTER RESOURCES (NGCR)
OPERATING SYSTEMS STANDARDS WORKING GROUP (OSSWG)

STEVEN L. HOWELL, EDITOR
UNDERWATER SYSTEMS DEPARTMENT

7 MAY 1990

NAVAL SURFACE WARFARE CENTER

Dahlgren, Virginia 22448-5000 • Silver Spring, Maryland 20903-5000

REPORT DOCUMENTATION PAGE

Form Approved OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	7 May 1990	

4. TITLE AND SUBTITLE
Evaluation Process Report for Next Generation Computer Resources Operating Systems Interface Baseline Selection

5. FUNDING NUMBERS

6. AUTHOR(S)
Steven L. Howell, Editor

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Naval Surface Warfare Center (U33)
10901 New Hampshire Avenue
Silver Spring, MD 20903-5000

8. PERFORMING ORGANIZATION REPORT NUMBER
NAVSWC TR 90-248

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Space and Naval Warfare Systems Command (SPAWAR 3243)
Washington, DC 20365-5109

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

The Operating Systems Standards Working Group (OSSWG) has been tasked to evolve an interface standard for operating systems. This document defines the process by which the OSSWG will make a recommendation of a baseline interface specification to the Next Generation Computer Resources (NGCR) program office. This baseline specification will be derived from one or more existing operating system implementations, specifications, or standards.

14. SUBJECT TERMS
NGCR OSSWG

15. NUMBER OF PAGES
53

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	SAR

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. Z39-18
298-102

NAVSWC TR 90-246

EVALUATION RESULTS REPORT FOR NEXT GENERATION COMPUTER RESOURCES OPERATING SYSTEMS INTERFACE BASELINE SELECTION

BY NEXT GENERATION COMPUTER RESOURCES (NGCR)
OPERATING SYSTEMS STANDARDS WORKING GROUP (OSSWG)

STEVEN L. HOWELL, EDITOR
UNDERWATER SYSTEMS DEPARTMENT

7 MAY 1990

NAVAL SURFACE WARFARE CENTER

Dahlgren, Virginia 22448-5000 • Silver Spring, Maryland 20903-5000

REPORT DOCUMENTATION PAGE

Form Approved
OMB No 0704-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	7 May 1990	

4. TITLE AND SUBTITLE
Evaluation Results Report for Next Generation Computer Resources Operating Systems Interface Baseline Selection

5. FUNDING NUMBERS

6. AUTHOR(S)
Steven L. Howell, Editor

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Naval Surface Warfare Center (U33)
10901 New Hampshire Avenue
Silver Spring, MD 20903-5000

8. PERFORMING ORGANIZATION REPORT NUMBER
NAVSWC TR 90-246

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Space and Naval Warfare Systems Command (SPAWAR 3243)
Washington, DC 20365-5109

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited

12b. DISTRIBUTION CODE

13. ABSTRACT (Maximum 200 words)
This report summarizes the results of the Next Generation Computer Resources (NGCR) Operating Systems Standards Working Group (OSSWG) evaluation of candidates for the Operating System Interface (OSIF) Baseline.

14. SUBJECT TERMS
NGCR OSSWG

15. NUMBER OF PAGES
170

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	SAR

NUSC Technical Document 6902
1 June 1990

Recommendation Report for the Next-Generation Computer Resources (NGCR) Operating Systems Interface Standard Baseline

Operating Systems Standards Working Group (OSSWG)
Compiled by D. P. Juttelstad (NUSC)

Naval Underwater Systems Center
Newport, Rhode Island New London, Connecticut

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	1 June 1990	Final

4. TITLE AND SUBTITLE
Recommendation Report for Next-Generation Computer Resources (NGCR) Operating Systems Interface Standard Baseline

5. FUNDING NUMBERS
PN A45146

6. AUTHOR(S)
Operating Systems Standards Working Group (OSSWG)

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Naval Underwater Systems Center
Newport Laboratory
Newport, RI 02841

8. PERFORMING ORGANIZATION REPORT NUMBER
TD 6902

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Space and Naval Warfare Systems Command
(SPAWAR-324)
Washington, DC 20363

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT (Maximum 200 words)

The Next-Generation Computer Resources (NGCR) Operating Systems Standards Working Group (OSSWG) conducted a survey of existing operating systems and operating systems interface standards to establish a baseline for the NGCR operating system interface. This report presents the results of that survey and the OSSWG recommendation for the standard baseline.

14. SUBJECT TERMS
Next-Generation Computer Resources
Operating Systems Interface

15. NUMBER OF PAGES
18

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	SAR

EXECUTIVE SUMMARY

The Next-Generation Computer Resources (NGCR) Operating Systems Standards Working Group (OSSWG) conducted a survey of existing operating systems and operating systems interface standards to establish a baseline for the NGCR operating system interface (OSIF). As a result of this survey, the total number of operating systems considered was reduced from 110 to 7, which then were formally evaluated. These seven were Alpha, ARTX, CRONUS, iRMX, Mach, ORKID, and POSIX.

The formal evaluation consisted of assessing the seven candidates against the requirements contained in the "NGCR OSSWG Requirements Document" (reference 1) and a set of eight programmatic issues. The numeric results of this evaluation identified three candidates as superior: Alpha, iRMX, and POSIX. To obtain a clear consensus of the OSSWG, an anonymous ballot was held that resulted in POSIX obtaining a 51-percent majority vote. Based on the results of the balloting, the NGCR OSSWG recommends POSIX be selected as the NGCR OSIF baseline. The working group also recommends that the Navy and OSSWG capitalize on the strengths of the other candidates, particularly Alpha and iRMX, in the continuing standards development.

NUSC Technical Document 6904
1 June 1990

After-Action Report for the Next-Generation Computer Resources (NGCR) Operating Systems Interface Standard Baseline Selection Process

Operating Systems Standards Working Group (OSSWG)
Compiled by J. T. Oblinger (NUSC)

Naval Underwater Systems Center
Newport, Rhode Island • New London, Connecticut

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	1 June 1990	Final

4. TITLE AND SUBTITLE
After-Action Report for the Next-Generation Computer Resources (NGCR) Operating Systems Interface Standard Baseline Selection Process

5. FUNDING NUMBERS
PN A45146

6. AUTHOR(S)
Operating Systems Standards Working Group

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Naval Underwater Systems Center
Newport Laboratory
Newport, RI 02841

8. PERFORMING ORGANIZATION REPORT NUMBER
TD 6904

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Space and Naval Warfare Systems Command
(SPAWAR-324)
Washington, DC 02841

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT (Maximum 200 words)

The Next-Generation Computer Resources (NGCR) Operating Systems Standards Working Group (OSSWG) conducted a survey of existing operating systems and operating systems interface standards to establish a baseline for the NGCR operating system interface. This report reviews the OSSWG evaluation process and discusses issues that caused difficulty to OSSWG in meeting its objectives.

14. SUBJECT TERMS
Next-Generation Computer Resources
Operating Systems Interface

15. NUMBER OF PAGES
27

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	SAR

EXECUTIVE SUMMARY

The Next-Generation Computer Resources (NGCR) Operating Systems Standards Working Group (OSSWG) conducted a survey of existing operating systems and operating systems interface standards to establish a baseline for the NGCR operating system interface (OSIF). As a result of this survey, the total number of operating systems considered was reduced from 110 to 7, and those final 7 were then formally evaluated. The formal evaluation consisted of assessing the seven candidates against the requirements contained in the "NGCR OSSWG Requirements Document" (reference 1) and a set of eight programmatic issues.

The first section of this report describes the purpose and scope of this study, which covered the timeframe from March 1989 (a briefing made to industry) to April 1990 (when the OSIF baseline was selected).

The second section discusses issues regarding the OSSWG evaluation process. Issues presented include the benefits OSSWG gained by active industry participation, the effectiveness of the electronic mail system for providing communications between meetings, the concerns about the compressed schedule, and a discussion about the difficulty in interpreting the evaluation scores.

The third section addresses the technical issues that caused difficulties for OSSWG in achieving its objectives. Some of these issues include (1) how to define distributed technology within an operating system interface; (2) how to specify security; (3) how security impacts the technology of real-time capabilities, distribution, and fault-tolerance; and (4) to what extent OSIF issues impact the performance of OS implementations. The technology topics in this section are presented as technology shortfall areas where there is need for additional research.

Reverse Blank

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1 AGENCY USE ONLY *(Leave blank)*	2 REPORT DATE	3 REPORT TYPE AND DATES COVERED
	April 1991	Final: Jan 89 – Sept 90

4 TITLE AND SUBTITLE

OPERATING SYSTEMS STANDARDS WORKING GROUP (OSSWG) NEXT GENERATION COMPUTER RESOURCES (NGCR) PROGRAM
First Annual Report—October 1990

5 FUNDING NUMBERS

PR: CC30
PE: 604574N
WU: DN587574

6 AUTHOR(S)

R. Bergman/Operating Systems Standards Working Group (OSSWG)

7 PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Naval Ocean Systems Center
San Diego, CA 92152–5000

8 PERFORMING ORGANIZATION REPORT NUMBER

NOSC TD 2101

9 SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Space and Naval Warfare Systems Command
Washington, DC 20363–5100

10 SPONSORING/MONITORING AGENCY REPORT NUMBER

11 SUPPLEMENTARY NOTES

12a DISTRIBUTION/AVAILABILITY STATEMENT

Approved for public release; distribution is unlimited.

12b DISTRIBUTION CODE

13 ABSTRACT *(Maximum 200 words)*

This document records the achievements of the Operating Systems Standards Working Group (OSSWG) from January 1989 – September 1990. The Next Generation Computer Resources (NGCR) Program of the U.S. Navy is seeking to establish standard interfaces of several types to provide an Open System Architecture for constructing Navy applications systems from compatible components. These interfaces are to be based on industry/commercial nonproprietary standards. Among the standard interfaces being sought is an operating system interface. The OSSWG was formed in 1989 to identify such a standard.

14 SUBJECT TERMS

operating systems
distributed systems
real-time systems
real-time distributed systems
mission-critical computer interfaces
standardized interfaces
computer-component interfaces

15 NUMBER OF PAGES

491

16 PRICE CODE

17 SECURITY CLASSIFICATION OF REPORT	18 SECURITY CLASSIFICATION OF THIS PAGE	19 SECURITY CLASSIFICATION OF ABSTRACT	20 LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	SAME AS REPORT

21a. NAME OF RESPONSIBLE INDIVIDUAL	21b. TELEPHONE *(include Area Code)*	21c. OFFICE SYMBOL
R. Bergman	(619) 553–4098	Code 412

INITIAL DISTRIBUTION

Code 0012	Patent Counsel	(1)
Code 0144	R. November	(1)
Code 412	R. Bergman	(15)
Code 952B	J. Puleo	(1)
Code 961	Archive/Stock	(6)
Code 964B	Library	(3)

Defense Technical Information Center
Alexandria, VA 22304-6145 (4)

NOSC Liaison Office
Washington, DC 20363-5100 (1)

Center for Naval Analyses
Alexandria, VA 22302-0268 (1)

Space & Naval Warfare Systems Command
Washington, DC 20363-5100 (10)