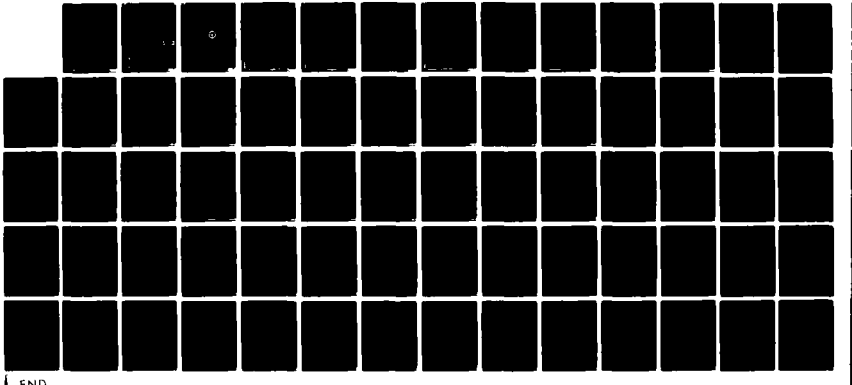
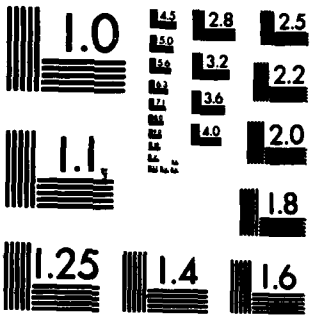
AD-A128 904 SOFTWARE TECHNOLOGY FOR ADAPTABLE RELIABLE SYSTEMS (STARS) FUNCTIONAL TASK AREA STRATEGY FOR SYSTEMS(U) OFFICE OF THE DEPUTY UNDER SECRETARY OF DEFENSE RESEARCH AND E .. 21 APR 83 1/1

UNCLASSIFIED F/G 9/2 NL



END
DATE FILMED
6-83
DTIC



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1 REPORT NUMBER	2 GOVT ACCESSION NO. / 3 RECIPIENT'S CATALOG NUMBER
4 TITLE (and Subtitle) Software Technology For Adaptable Reliable Systems (STARS) Functional Task Area Strategy For SYSTEMS	5 TYPE OF REPORT & PERIOD COVERED / 6 PERFORMING ORG. REPORT NUMBER
7 AUTHOR(s) The DoD Joint Service Task Force on the Software Initiative (STARS)	8 CONTRACT OR GRANT NUMBER(s)
9 PERFORMING ORGANIZATION NAME AND ADDRESS AD A128904	10 PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11 CONTROLLING OFFICE NAME AND ADDRESS Deputy Under Secretary of Defense Research & Advanced Technology Washington, D.C. 20301	12 REPORT DATE April 21, 1983 / 13 NUMBER OF PAGES 66
14 MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)	15 SECURITY CLASS. (of this report) Unclassified / 15a DECLASSIFICATION/DOWNGRADING SCHEDULE

16 DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited

17 DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

Unclassified

18 SUPPLEMENTARY NOTES

19 KEY WORDS (Continue on reverse side if necessary and identify by block number)

Ada, STARS, embedded computers. KIT, KITTIA, automated support environments, system interface standards, standardazation, mission critical systems, military systems, software, hardware, software initiative,deliverables, Reliablity, VHSIC , software engineering institute.

20 ABSTRACT (Continue on reverse side if necessary and identify by block number)

This strategy document identifies the scope, sub-objectives and strategies designed to provide the conceptual appraoch for accomplishment of the STARS program objectives in the systems funcitonal task area. It identifies and describes the high-level activities, products and capabilities. In order to provide full understanding, background and rationale material is sometimes covered that is also in STARS Program Strategy.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE S/N 0102-LF-014-6601

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

SOFTWARE TECHNOLOGY FOR

ADAPTABLE, RELIABLE SYSTEMS (STARS)

FUNCTIONAL TASK AREA STRATEGY FOR

SYSTEMS

Department of Defense

21 April 1983

FOREWORD

This strategy document is one of eight functional task area strategies produced by the STARS Joint Task Force. All of the documents produced by the Task Force, including the general STARS Program Strategy document, are listed in the STARS Joint Task Force Report.

This document identifies the scope, sub-objectives and strategies designed to provide the conceptual approach for accomplishment of the STARS Program objectives in the systems functional task area. It identifies and describes the high-level activities, products and capabilities. In order to provide full understanding, background and rationale material is sometimes covered that is also in STARS Program Strategy.

These functional task area strategy documents do not attempt to delineate the detailed plans, costs and procedures for bringing the proposed products and capabilities into being and do not identify the form of the particular projects that will undertake the work nor the organizations in which the work will be accomplished. Instead, these strategies are intended to guide the process of such implementation planning and accomplishment.

Indeed, because of the high degree of linkage among the functional task areas, implementation plans and acquisitions may well combine related capabilities and products across areas. Individual projects may tackle only part of one subtask from a functional area or several subtasks from several functional areas.

Thus, this functional task area strategy describes broad, achievable requirements for accomplishing the relevant STARS objectives. Its main purpose is to help guide the implementation planning process.

TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

1.0 OVERVIEW

Software is only one part of DoD mission critical computer systems, and these systems must be addressed from an overall systems point of view. The Systems area is concerned mainly with the target system environment but includes some concern with its relationship to its support system environment. A target system environment is the configuration of systems software and hardware in which the applications software operates. The Systems area addresses the issues related to systems software and hardware. It is responsible for providing access to the systems technology base and advancing it in response to expected future mission needs. Improvements in the overall quality of defense systems depends upon a corresponding increase in the quality of the underlying systems software and hardware. This in turn requires that methods, tools, and knowledge to make effective use of the advanced systems technology be developed and placed in the support systems environment.

This document presents the scope, objectives, strategy, and general activities for the Systems Functional Task Area. The first section is a general overview of the area. It describes the relationship of the Systems area to activities outside STARS and to other STARS areas. In addition, the general overview describes the scope, objectives, and strategy in terms of the consolidation, enhancement, and transition phases. Following the general overview section are sections for each phase: consolidation, enhancement, and transition. Each phase section gives a more specific overview and describes the scope, objectives, and strategy in terms of the kinds of activities for which plans are needed.

1.1 Scope

The Systems area is concerned with target systems, their enhancement, and the means by which they can be effectively used. A target system is a configuration of hardware and systems software

that provides the target systems environment in which the application software developed in a support system environment operates. Early in STARS the support system environment is only concerned with the development of software. However in the future, systems including hardware may also be designed using the support system environment.

From the point of view of a system developer, applications specific software (or hardware) is developed for configuration with a target system that is itself a configuration of systems capabilities generally packaged in modules.

Systems capabilities are the most generic since they are designed to be used for a variety of applications. They tend to be the most modular so that (1) the services required by each application can be effectively configured, (2) resources can be conserved by including only those modules needed, (3) performance may be tuned, (4) realtime and resource constraints can be satisfied, and (5) the exploration of new technologies may be accomplished in a timely fashion. In the future, such systems capabilities may be provided by hardware and software combinations very different from the traditional computer systems in common use today.

DoD mission needs will require improvements in the quality of DoD computerized systems because of their growing pervasiveness and the increasingly critical "responsibilities" allocated to the computer software and hardware. These needed quality upgrades will involve a number of properties; particularly important are adaptability and reliability, as reflected by their inclusion in the name of the STARS program. Consideration of these properties plays a key role in the objectives and strategy of the Systems Functional Task Area, as will be discussed in the following sections.

Four topics that appear to provide the greatest benefit have been identified, with the realization that these may be broadened in

the future. These are systems architecture, systems software, software/hardware synergy, and environmental concerns.

Systems architecture is emphasized because new architectures (such as distributed, functional, and data flow architectures) hold significant promise for innovative approaches to systems. However, much more needs to be done on both the applicability of the architecture to DoD problems and providing access to them through support systems environments. In addition, new architectures will provide the means by which target system packages and their configuration can be more adaptable and reliable with higher functionality.

Systems software is emphasized because it is the means by which adaptability can be provided in a configuration of systems packages. It bridges the gap from higher-level systems functions to the underlying hardware. In the long term, however, a blending of systems architecture and systems software issues may occur, making the distinction between them less meaningful.

Software/hardware synergy is emphasized because the expected rapid advancement of both software and hardware technology over the next decade raises many questions about how to design systems. The recent emergence of VHSIC/VLSI technology raises the question of which system parts should be implemented in software and which parts in hardware. Of particular interest are methods, tools and knowledge that assist in the co-evolution of software and hardware.

Environmental concerns are emphasized because of the importance of the relationship between the target system environment and the support system environment throughout the development and useful life of a system.

The Systems area should cooperate with the software and hardware systems technology activities external to STARS. The development of new systems hardware technology and, to some extent, systems software

technology is outside the scope of STARS. Hardware systems technologies are clearly the responsibility of external activities, although STARS has a responsibility to make its needs known.

The Systems area will couple to and complement the potential of advances in software and hardware technology in other activities. Among the activities of interest are Ada with its environments, applications of Ada to produce systems software, the DoD VHSIC program, the DARPA VLSI systems and Supercomputer programs, the DoD Computer Security Program, the NASA Reliability Program, and industry breakthroughs in microprocessors and their applications. The areas of interest include those that provide generic system functionality such as operating systems and database management systems with emphasis on reliable performance in realtime on distributed architectures for a variety of applications with changing requirements.

The Systems area addresses the improvement of the target system and will include developing tools and methods that will eventually reside in the support environment developed by the Support Systems area. Techniques developed to improve "systems" may also later be applied to application and support systems. In addition, developments in the Support Systems area will surely help the quality of systems software built using the support environment. One key difference between the Systems area and the Support Systems area is that Systems focuses on improving the target system itself while Support Systems focuses on improving how such systems are developed and supported. Another key difference is that Support Systems is responsible for final integration, delivery, and support of tools and methods.

1.2 Objectives

The objective of the Systems area is to provide the basis for the "systems"-related capabilities needed to meet the requirements

for future DoD mission critical target systems. This implies that for future DoD projects we must be able to provide the required levels of the following kinds of properties:

- o Performance (adequate for realtime applications)
- o Adaptability (adequate for changing mission needs and new technologies)
- o Reliability (adequate for different mission lengths)
- o Security (adequate for perceived threats against protected objects)
- o Fault Tolerance (adequate for technology failures)
- o Survivability (adequate for perceived mission threats)
- o Maintainability (in the logistic sense)
- o Affordable (for value of the mission supported)
- o Timely Construction (to be available for use when needed).

Although this may not be a complete set of the properties that embody the notion of higher quality mission critical systems, it does represent some areas in which improvement appears to be needed. Their identification, quantification, and achievement should provide a basis for quantifying success in the Systems area. The STARS System area must measurably increase the Nation's ability to design, build, field, and support DoD mission critical systems having these kinds of properties.

This task area should contribute to meeting the challenge of increasing the power of application-independent tools, especially for the development and support of complex systems. In addition, this area should produce more powerful tools and methods for using the innovative computer systems architecture made possible by the VHSIC, VLSI, and Supercomputer programs. Thus, the Systems area concerns, which directly include three terms from the STARS program name --

adaptability, reliability and systems -- are key to achieving the very substantial improvements that are needed for future DoD systems.

1.3 Strategy

The objective of meeting the needs of future mission critical systems implies that their requirements must be analyzed to determine the necessary levels of the various properties. Note that a future system will require a set of values covering all the properties -- values which may be harder to achieve because they must exist together. For example, high performance and adaptability or reliability have often been difficult to achieve simultaneously.

The central strategy for the Systems area is:

- Review DoD mission development plans and analyze planned future systems, including major upgrades, to derive the property combinations needed over time.
- Use these needed property combinations (and the evaluation criteria developed for them) to drive R&D and inform the marketplace.
- Prototype, assess, demonstrate, and tool the results along with other parts of STARS to make them suitable and attractive for use.
- Make results available for wide (re)use.

  Use of the result meets real needs since that is what they have been aimed at from the beginning.

This strategy should not be a one shot exercise, but rather should be repeated regularly. Future needs should be periodically reanalyzed as plans change or become clearer. These new analyses should then be used to redirect efforts. This continuing process should result in continuing progress properly focused on meeting DoD future mission requirements.

The more specific strategy and activities in the Systems area will be described using the three STARS phases -- consolidation (3 years), enhancement (3 years), and transition. The strategy given for the consolidation phase is based on the current understanding of future needs and technology opportunities and should evolve as the understanding of future needs evolves. Naturally, the strategy is more specific in the near term and more general in the long term.

This section presents an overview of the strategy for each phase of the Systems area. A more specific strategy for each phase and the kind of activities for which plans are needed is presented in the following section.

The Systems Area Consolidation Phase builds upon the systems technology base, mission needs, and research results to produce a more powerful, consolidated, and uniform Systems area. A more consolidated systems technology base accessible through Ada and its support environment should improve the state of practice for major upgrades and new developments of mission critical systems using the available systems technology. Ada should be used as a vehicle in the consolidation phase and not as an end in itself for the long term. The assessment of mission needs will be useful in identifying and quantifying the Systems area objective. An assessment of existing research results in the Systems area will be useful in identifying approach that should be used in new-systems developments and in identifying areas where research is required. Among the areas to be addressed should be

- computer architecture including distributed systems
- systems software including a widely usable set of realtime operating system interfaces
- software-hardware synergy including co-design

- o reliability, including testing, fault tolerance, and formal methods of specification and verification
- o environmental concerns, including transfer of results to the Support Systems automated environment.

The results of the consolidation phase must provide not only usable results but also a basis for the enhancement phase.

The Systems Area Enhancement Phase builds upon the results of consolidation by pursuing both evolutionary and revolutionary paths. Enhancement in the systems area includes improving DoD's mission system assessment capability, improving the focus of research on mission system needs, and accelerating technology insertion of these results. Cooperation with external activities during the consolidation phase should begin to produce results that may be incorporated into the Systems area activities. Emphasis should be placed on developing effective ways to enhance the systems technology base so that it will be prepared to satisfy future mission system needs.

The Systems Area Transition Phase will institutionalize the results and process of enhancement so that the systems technology base will continue to progress and continue to remain prepared to satisfy mission system needs after the STARS program is over. The result of transition is that DoD will be doing accurate mission assessment, effective focusing of R&D efforts, and accelerated technology insertion.

The Systems area is very broad and STARS resources are limited. So, in addition to carefully targeting its efforts, the Systems area must try to inspire others by articulating its needs and must exploit advances made elsewhere, particularly in hardware and computer systems architecture. The Systems area should serve as a conduit for delivering results to the systems community. It can do this by cooperating with related activities outside STARS and incorporating

their developments into advanced system technologies (including related tools) for delivery to the Support Systems area.

Altogether, the Systems area is a foundation of STARS in that it improves the underlying systems technology which will be used in future systems. The systems developed using the Systems area results should directly demonstrate progress made toward achieving STARS objectives.

2.0 DETAILED STRATEGY

The consolidation, transition, and enhancement phases of the STARS Systems area are described in more specific terms. For consolidation, the subareas or subtasks are described. For enhancement, the scope, objectives, and strategy are given followed by a summary of the activities for which plans are needed. The enhancement phase is more research-oriented than the consolidation phase. For transition, only general strategy can be addressed at this time.

2.1 Consolidation Phase

The Consolidation Phase will start a number of activities, each building on currently existing or planned activities outside STARS. These include:

- Start the use of modern system engineering through Ada
- Start mission systems needs assessment to establish STARS System objectives
- Start R&D systems focus on needs and demonstrations
- Start identifying technology insertion opportunities.

Ada with its early support system environment should be used as a vehicle for improving the state of practice. In addition, mission assessment, research focus, and preparation for technology insertion issues are included.

The Strategy for the STARS Systems Area Consolidation Phase will improve the state of practice in systems directed toward mission needs. It includes near-term usable results and preparation for the Enhancement Phase. Production of usable products, development, and research will all be conducted.

The state of practice of systems technology may be improved by providing access to systems technology through a modern programming language with an integrated support system environment. Ada with its

early environment should be a useful vehicle for achieving this objective. Activities in this area should provide the means to start using Ada for systems development as early as possibly without requiring that existing systems software be rewritten in Ada. This will allow existing and developing systems software to be used through Ada interfaces.

The state of practice of mission assessment should be improved by focusing analysis of mission development plans for system needs within a STARS Systems mission assessment activity. The identification of a set of DoD system technology needs would be a useful result of consolidation that would focus attention during enhancement.

The ability to focus research on mission system needs should be improved by identifying systems problems and criteria for evaluating proposed solutions. Activities designed to demonstrate experimentally a proposed solution in prototype form should provide a useful assessment of some existing results, identify those that might be applied, and point to areas where more research is needed.

The state of practice of technology insertion for mission system needs should be improved by quantifying mission system needs and identifying technology insertion opportunities. Plans for technology insertion should be documented with their evaluation criteria.

The combination of these kinds of activities should encourage the communities of interest to work together toward improving the state of practice in the systems area.

The suggested activities within each subtask are described at some length for the consolidation phase. These provide a specific description that helps explain what is needed. It is not expected that they will all be performed, or that when performed, they will exactly follow the details given here.

2.1.1 Computer Architecture Subtask

The purpose of the architecture subtask is to support the DoD systems designer by:

(1) aiding in the selection of computer architectures for embedded computer systems with increased performance, adaptability, reliability, and other properties

(2) providing methods for designing and programming systems with innovative architectures,

(3) providing tools that support these methods,

(4) providing standardized system component interfaces in order to simplify the integration of a system,

(5) providing methods and tools for reusing software on different hardware configurations.

While this subtask starts in the consolidation phase, similar activities should continue throughout the STARS program. The strategy for this subtask looks at three successive areas of emphasis: early, where the emphasis is on the evaluation of architectures that support Ada; middle, where the emphasis is on methods and tools for designing and programming loosely-coupled, heterogeneous, distributed systems; and later, where the emphasis is on methods and tools for designing and programming nontraditional architectures, such as tightly-coupled, homogeneous machines, data flow machines, and functional language machines.

The subtask consists of five activities:

(1) developing techniques for performing a cost-benefit analysis of this subtask,

(2) evaluating architectures that support Ada in terms of their ability to provide performance, adaptability, reliability, and other properties

(3) developing standard interface definitions to allow quick, reliable, and efficient construction of systems adapted for

particular problems from a common base of hardware components and interconnection protocols and a common base of configurable systems software components,

(4) developing tools to aid the designer in decomposing problems to run on a distributed system, managing a distributed system (e.g. scheduling tasks or assigning resources), and constructing a distributed system through the use of standard protocols and reusable software for commonly required functions,

(5) evaluating innovative architectures (such as tightly coupled homogeneous systems, data flow machines, functional language machines) in terms of their ability to provide performance, adaptability, reliability, and other properties.

2.1.1.1 Assessment of Architecture Subtask. The purpose of this activity is to develop methods for performing cost-benefit analyses of this subtask. It would provide a means for estimating the cost savings generated by the subtask by measuring the impact of the architecture subtask work, e.g. how widely the tools developed are used, and by estimating the cost savings for those projects that are affected. The implementation and use of these methods would be done by the Measurement functional task area.

Description of Projects

The first project would define the measures to be used for determining the benefits of the architecture subtask. Benefits to be measured include cost, mission impact (i.e., how important is an aspect of a system to the performance of the mission), and development time. Measures could include potential benefits of a project or activity, impact of project on DoD systems (e.g., how widely a design method is actually used), and estimates of the cost of systems if the project or activity was not done.

Once the measures have been defined, the second project would define methods for collecting data to evaluate the activities or projects using the measures.

And the third project would determine methods for estimating the cost of DoD embedded computer systems (ECS) development and support in the absence of the methods and tools generated by this subtask, and compare those costs with the costs of this subtask.

Deliverables

(1) Benefit measure definitions.

(2) Methods for collecting data on use of methods, tools, and techniques.

(3) Methods for estimating cost avoidance and other benefits due to the use of methods, tools, and techniques.

Costs and Benefits

This activity is essential for the monitoring the impact of the subtask. It requires a modest investment, but could be important in guiding the implementation of the subtask by providing feedback on the impact of various projects.

2.1.1.2 Evaluate Ada Architectures. This activity would evaluate architectures to see how well they support software systems written in Ada. By architecture, we mean the virtual target machine that is experienced by the Ada programmer, including the impacts of the hardware, the operating system, the run-time support library, and the compiler.

Description of Projects

The first project would define comparison criteria such as measures of performance, reliability, memory capacity, address space, power consumption, and weight of the hardware; the speed, size of code generated, and correctness of the compiler; and the performance, flexibility, and security of the operating system. A great deal of work has already gone on in the area of architecture evaluation. The

purpose of this project is to collect these measures and integrate them into a useful evaluation package.

The second project would take the evaluation criteria determined and construct tools for collecting the data required by the measures such as benchmark programs, checklists, and surveys. Some evaluation should be performed by other established organizations, such as the evaluation of security by the Computer Security Center. This project should set up referral procedures and procedures for collecting data from other evaluation projects.

The third project will distribute the comparison criteria to provide systems designers with data that they can use to select system components, and to encourage the development of architectures that meet DoD needs.

Lastly, following the example of the Computer Security Center, a project might encourage manufacturers to submit their systems for evaluation. Once the manufacturer has agreed to an evaluation, the results should be published no matter how they turn out.

Deliverables

The first deliverable of this activity is a document defining the comparison criteria for the Ada architectures. The second deliverable is a set of tools for testing Ada architectures to generate comparison metrics. The third deliverable is a set of comparisons of systems based on the comparison procedures.

Costs and Benefits

The costs of evaluating Ada systems should be comparable to the costs of validating compilers (which is being done by the National Bureau of Standards and the Ada Joint Program Office) or evaluating the security level of operating systems and their supporting hardware (which is being done by the Computer Security Center). Extrapolating

from the experience of the latter the development of the comparison criteria is a difficult and expensive task.

The first two deliverables would benefit Ada system developers by giving them ground-rules for comparison, so that they can evaluate their system at different stages in its development. The third deliverable would benefit the DoD system designer, since it could provide a basis for comparison shopping for system components.

2.1.1.3 Design Standard Interfaces. The purpose of this activity is to develop standard interface definitions that provide a basis for system integration from standard components, and (as a result) for the concurrent co-design of software and hardware starting from a common fixed basis. Standards for many system components already exist (such as busses, terminals, disks); this task will integrate these standards and develop new standards (particularly for intelligent peripherals) so that systems can be developed that take advantage of VLSI, yet still can be incorporated in DoD embedded computer systems with minimal additional engineering costs due to interface problems. Examples of areas where intelligent peripherals are developing include graphics, where entire picture processing systems can be attached to a processor; database management, where intelligent disk units can directly execute queries; and in signal processing, where special purpose processors attached to a control unit as peripherals perform mathematical functions such as Fourier transforms. The increased intelligence of these peripherals makes the interfacing problems more difficult because there are so many more options. The interfaces must be defined in a manner compatible with the systems software interfaces being defined and the developing concepts for software-hardware synergy.

Description of Projects

The first project would select an initial collection of modules which would then have their interfaces designed. The modules should be selected to improve the software-hardware synergy of applications identified by the review of future plans and by the applications specific area. The modules should also be selected to support and gracefully fit with the systems software task. If acceptable interface definitions already exist, then these could be noted and included.

The initial specifications of the modules would be a composite of English text, programs, and formal descriptions to provide a basis for their discussion.

When the initial specifications have been prepared, the next project should have them reviewed by users and developers. The result would be a revised set of specifications.

Once the informal module specifications are frozen, a project would construct tools for using the specifications. The first set of tools might consist of executable versions of the specifications, together with test programs that can be used to verify that an implementation of a module meets its specifications or that the system that uses the module meets the interface specifications. These executable versions of the specifications should be written in an acceptable hardware description language or a language defined by the software-hardware synergy subtask.

In the next project, formal specifications would be developed for selected modules in order to support efforts by the software-hardware synergy and reliability subtasks for formal verification of systems. This project would supply formal descriptions of the modules that can be included in the formal description of the total system and manipulated by the automated tools to check the consistency of the total system description.

The final project of this activity would be the demonstration of the utility of interface definitions. These demonstrations might be constructed by the systems software subtask (for modules that are required by a realtime operating system or a realtime database management system) or as part of a demonstration of software-hardware synergy methods.

Deliverables

The deliverables for the interface definition subtask would be documents, executable definitions, and formal (e.g., axiomatic) specifications of standard interfaces between system software and hardware components such as intelligent peripherals. These interfaces will provide a common point of reference for software and hardware co-design. For example, the definition of a set of standard devices will serve as the basis for the development of the device interfaces (drivers) for a portable and generic operating system, which will be developed by the Systems Software subtask.

Costs and Benefits

Cost can be derived from a cost for informal specifications and for executable definitions of each of the interfaces, plus a cost for formal specifications for some interfaces.

The benefits of this activity include reduced design costs and development time of systems because compatible interface definitions already exist, increased reliability of systems because many interface definitions have already been tested, and reduced costs for system evolution through the use of plug-compatible component upgrades.

2.1.1.4 Distributed Systems Design Methods and Tools. The purpose of this activity is to develop tools for designing and programming distributed systems. The emphasis of this activity is on loosely coupled, heterogeneous systems. Another activity evaluating

innovative architectures would examine more advanced types of distributed architectures.

Description of Projects

One project would address communications protocols, which are a critical part of every distributed system. One or more higher levels of communications protocols are needed to support simple message transfer, message exchange, or extended "conversations" among tasks. The goal of this task would be to encourage development of a set of protocols to meet different needs. For example, it is probable that different protocols would be needed for distributed task allocation, secure message transmission, and the handling of messages with multiple priority classes.

Another project would develop models for distributed computations that could: define where authority and responsibility resides for different classes of actions, explain when such authority is transferred between sites, define what inconsistencies could arise among tasks, and indicate how the tasks would cope with these inconsistencies. These models should be integrated with the methods developed by the software-hardware synergy subtask.

A project would address techniques for mapping applications onto distributed systems by decomposing the application into appropriate modules and allocating the modules to the processors of the network. This project should collect information about natural and effective decompositions and techniques for both determining and evaluating alternative decompositions. Currently, systems are often decomposed along simple boundaries which reflect special-purpose hardware. Future distributed systems will require a more subtle decomposition, especially in the case of applications which use cooperative problem solving.

A project might be launched to develop techniques for reconfiguring systems in order to increase the responsiveness of the system to changing loads and to increase the reliability of the system. This could include ways of balancing processing and bandwidth loads (both statically and dynamically) ways of routing information around a defective link or node in order to provide improved system reliability.

The final project would encourage the use of the methods and tools developed by this activity in the design and construction of distributed systems. The strategy for this project could be to find distributed systems that are planned for construction and to provide additional funds for these projects to support training in the use of the tools and collection of data about the design, construction, and maintenance of the system. Substantial progress in the area of distributed systems can probably only be assured by building a varied collection of systems, based on different models and used for a variety of applications and in a variety of situations. It takes roughly 5 years to build a substantial system and 8 to 10 years before it works effectively. Progress could be made most rapidly if a number of technologically interesting systems are constructed in parallel.

Deliverables

The deliverables for this activity are methods and tools for constructing distributed systems, and prototype systems developed using these methods and tools.

Costs and Benefits

Cost estimates can be based on a breakdown of protocol definition, models, partitioning, reconfiguring, and software conversion, plus training and support during the development of several prototype systems.

The analysis of benefits of this activity depends upon three factors: the importance of distributed systems in DoD ECS, the impact of this work on improving the design of distributed systems, and the breadth of use of the methods and tools developed by this activity.

Distributed systems are essential to military systems, for survivability, for support of geographically diverse functions, and for performance. Furthermore, the advantages of VHSIC/VLSI (i.e. the ability to generate many copies of a circuit in a small space and at little cost) naturally lead to distributed system architectures as a solution to the performance limits imposed by the physics of the devices. Thus the potential for payoffs in the development of distributed systems is very large.

2.1.1.5 Evaluation of Innovative Architectures. This activity would examine innovative architectures to determine their long range impact on DoD ECS and to examine ways of minimizing the cost of developing major software bases for the architectures. Architectures to be considered should include tightly coupled distributed architectures, data flow architectures, functional architectures, and inference architectures. This would be a long lasting project; its goal is to monitor the development of architectures so that the methods and tools developed under STARS would be applicable to the new architectures. This subtask needs to be closely coordinated with the VHSIC program and the DARPA VLSI and Supercomputer programs as well as other outside efforts.

The first project would identify classes of architectures with similar areas of application, similar capabilities, and similar limitations. By identifying the classes, the process of tracking their development and encouraging research in particular areas should be simplified.

Next a project would define ways of comparing the capabilities of the architectures. The criteria and ways to measure the ability of an architecture to meet the criteria could be published, so that researchers looking at architectures would have goals and benchmarks for evaluating their architectures. This would indirectly encourage architects to design systems that are responsive to DoD needs.

Another project would be to identify architectures of particular promise for particular types of problems. This project could sponsor studies comparing architectures using the evaluation criteria developed.

Finally a project would look at ways of using available software to generate software for the new architectures. Many of the more unusual architectures are not designed to support Ada, and most of the parallel processing architectures will not perform well if their software was designed to execute on a uniprocessor. Therefore, methods and tools would be useful that would allow the system designer to convert a software system developed in a different environment to operate effectively on an innovative architecture.

Deliverables

The first deliverable of this activity would be a document describing the classes of architectures that will be examined. The second deliverable would be a document describing the criteria for evaluating the architectures including the performance, reliability, and difficulty of constructing a powerful software base for the architecture. The third deliverable would be a document evaluating selected architectures. The fourth deliverable would be a set of methods for constructing software bases for the architectures by such means as porting software to the system.

Costs and Benefits

Cost figures can be based on providing modest resources (say, one person-year) for each of the projects except the last, which should require more.

The benefits of this activity are a basis for understanding the impact of innovative computer architectures that have been or will be developed, together with tools for providing software for these innovative architectures.

2.1.1.6 Interactions with Other Efforts.

DARPA Super Computer and VLSI Programs

This work could support the architecture subtask by providing innovative architectures and components for evaluation. The architecture subtask would support this effort by providing evaluation criteria for comparing these architectures.

VHSIC

The architecture subtask could use the hardware description tools for defining component interfaces. It would support the VHSIC program by supporting the development of architectures that are based on chips developed under the VHSIC program.

ONR VLSI and Special Purpose Architecture Program

The Office of Naval Research is sponsoring work on special purpose architectures, particularly for signal and image processing. This work could support the architecture subtask by providing innovative architectures for evaluation. In combination with the application task, the architecture subtask could define Ada packages and hardware interface definitions, such as a linear algebra package or a signal processing package, that describe how to interface high-performance special-purpose machines designed for the ONR projects with other components of DoD embedded computer systems.

Applications Specific Area

In order to focus attention on the development of cost-effective architectures for tasks of interest to DoD, this subtask will focus on architectures that solve problems in application areas that are identified by the Applications task area. There should be frequent exchanges of information between the architecture subtask and the applications task to define the interfaces between software (i.e. the Ada packages) and hardware (e.g. special purpose architectures) in ways that promote software/hardware synergy.

Software Hardware Synergy Subtask

The architecture subtask would use the evaluation techniques developed by the software-hardware synergy subtask to evaluate architectures designed by the architecture subtask. The architecture subtask would support the use of the design tools and methods developed by the software-hardware synergy subtask.

Reliability Subtask

The architecture subtask would research the architecture of highly reliable distributed systems. An area of emphasis could be the software-hardware interface, e.g., software control of recovery and reconfiguration. The architecture subtask would depend on the reliability subtask to define methods and tools for designing such systems, and for verifying that the systems meet the reliability requirements.

2.1.2 System Software Subtask

The goals of the system software subtask are to design generic, Ada-based system software that can be used in a wide variety of DoD embedded computer systems (ECS), and to support the implementation of several versions of the software. The unifying concept behind this subtask is that of a family of virtual machines that are customized

for different applications by assembling different standardized hardware components and different system software components. Two key demonstrations of this concept are to be constructed: a realtime operating system, and a real-time database management system for large databases. These demonstrations will not only verify the effectiveness of the approach, but will provide useful software both for embedded computer systems (which is the primary emphasis) and for support systems (which is a secondary emphasis). For example, the realtime database management system will be designed for such applications as realtime target identification, where the signatures of different types of targets are stored and retrieved. However, much of the same software could be used for a software configuration management system. Another example would be a virtual machine to support realtime graphics. It would be designed for supporting such applications as a heads-up display, but could also support the graphics of the workstation being developed by the Human Engineering area.

The strategy for developing the family of virtual machines is based on the idea of a functional interfaces for conceptual (virtual) machines. The lowest level might be a "bare-bones" machine. A higher level virtual machine is constructed by adding functions that make use of available functions of the machine. The key design decisions are defining the units (and possibly layers) of functionality so that the lower levels can be shared by a wide variety of applications.

System software is the interface between the computer architecture, which is being studied by another subtask in the systems area, and the applications software, which is being developed by the applications-specific area. Thus this subtask will interact with both of these tasks. The strategy for this effort is a top-down design, but a bottom-up implementation. This means, for example, that generic device drivers would be implemented and available before

the system is complete, and can be used in other projects by DoD contractors who cannot wait for the complete system, or who cannot use the final system. The other part of the strategy could be to develop support tools in parallel with the system. These support tools would attempt to support the construction of specific members of the family of systems, just as an application program generator or composition systems constructs a member of a family of application programs.

2.1.2.1 Assessment of Systems Software Subtask. The purpose of this activity is to develop methods for performing cost-benefit analyses of this subtask. It will provide a means for estimating the cost savings generated by the subtask by measuring the impact of the systems software subtask work, e.g. how widely the tools developed are used, and by estimating the cost savings for those projects that are effected. The implementation and use of these methods will be done by the measurement area.

Description of Projects

A project would define the measures to be used for determining the benefits of the systems software subtask. Measures could include potential benefits of a project or activity, impact of project on DoD systems, and estimates of the cost of systems if the project or activity was not done.

Once the measures have been defined, a project would define methods for collecting data to evaluate the activities or programs using the measures.

Finally a project would determine methods for estimating the cost of DoD ECS development in the absence of the methods and tools generated by this subtask, and compare those costs with the costs of this subtask.

Deliverables

(1) Benefit measure definitions.

(2) Methods for collecting data on use of methods, tools, and techniques.

(3) Methods for estimating cost savings due to the use of methods, tools, and techniques.

Costs and Benefits

This activity is important for the monitoring the impact of the subtask. It requires a modest investment, but will be important in guiding the implementation of the subtask by providing feedback on the impact of various projects.

2.1.2.2 Define the Family of Interfaces. The purpose of this activity would be to develop an architecture for the family of common interfaces. The need for system software is universal, but the requirements made on the system software vary from application to application. This activity will provide a framework for extending the collection of system software modules in a consistent fashion. Systems constructed using this virtual machine approach could potentially be assembled from standard hardware components (i.e. components whose interfaces are specified by the architecture subtask) and from reusable software modules.

Description of Projects

In order to ensure that the family of interfaces would have enough flexibility to meet the needs of a variety of DoD applications, the first project will select an initial sample set of applications that will be used to evaluate the design and will provide a range of requirements. This project would define the range of requirements for the family of interfaces and determine the overlap in "systems" functions needed by the application. This project would

rely on the initial efforts of the subtask on focusing R&D on future DoD mission needs.

The next project would determine the modules of the system and show how the requirements of the applications identified previously could be met by members of the family of interfaces or virtual machines.

Then a project would construct plans for the development of members of the system. It would define specifications for the modules, and recommend an order of development that ensures that a basic system which could be generally useful is available early, and that customized members of the family would be developed later.

Lastly, a project might develop support tools for implementing the family of virtual machines. Included in the tools could be procedures for bootstrapping a portable Ada compiler, tools for generating specific device drivers from the generic device drivers defined by the architecture subtask, and tools for generating a particular configuration of the virtual system.

Deliverables

(1) The architecture of a family of interfaces (virtual machines).

(2) A plan for the development of the family which ensures that the most widely usable members of the family are developed first.

(3) Tools for constructing them.

Cost and Benefits

The cost of this activity would be modest. The benefit of this activity is a detailed technical plan for generating a variety of virtual machines, together with a plan for building these systems in the best order for widespread use.

2.1.2.3 Build and Demo, a Realtime Operating System. The purpose of this activity is to demonstrate the applicability of the virtual machine ideas developed in the previous activity by building a realtime operating system using the interfaces.

Description of Projects

In order to provide a convincing demonstration, an application in one of the areas selected by the Applications Specific area would be picked to act as the showcase for the demonstration.

After selection of the application the next project would take the description of the virtual machine modules prepared by the previous activity and design the realtime operating system by configuring modules.

Then a project would also take the descriptions of the modules determined by the previous task and generate detailed specifications and test and integration plans. At this point, many of the decisions about which functions are performed by the software, the hardware, and the firmware would be made.

A project would implement the first level of the system above the hardware-firmware level.

Then a project would implement the higher levels of the system, particularly those which are not likely to be immediately used in other systems, i.e., those functions that are essential support services that are peculiar to the demonstration system.

The demonstration of the operating system would require the use of the operating system by applications software. The final project would support the integration of the system software and the applications software to complete the demonstration system and demonstrate it.

Deliverables

The deliverable for this activity would be a working example of a realtime operating system designed in accordance with the virtual machine concepts and plan. This operating system would be applicable a wide range of applications.

Cost and Benefits

While the cost might be significant, several realtime operating system efforts are already in Service plans. The benefits of a demonstrated means of generating a family of realtime operating systems each of which is tailored to its application would be very large. It can be measured by computing the amount of time projected to be spent on realtime operating systems for the DoD, less the cost of generating appropriate members of the family. It is unreasonable to expect that all DoD realtime operating systems could be constructed in such a manner, but the costs will be recouped even if only a few are constructed in such a fashion.

2.1.2.4 Develop a Realtime Database System. The purpose of this activity would be to demonstrate the concepts of the family of virtual machines interfaces by constructing a realtime database management system (DBMS).

Description of Projects

In order to provide a convincing demonstration, an application in one of the areas selected by the Applications Specific area might be picked to act as the showcase for the demonstration.

Then a project will design the needed information structures. Realtime database management systems for DoD ECS applications pose special problems, including the presence of non-formatted data such as sensor data and images. They also may require data structures that deal explicitly with time, e.g., information on missile tracks

where the time of the last update is crucial and images that are too old need to be purged.

Then a project would take the description of the virtual machine modules prepared by the previous activity and design the database management system by selecting and interfacing modules.

Next a project would take the descriptions of the modules determined by the previous task and generate detailed specifications and test and integration plans. At this point, many of the decisions about which functions are performed by the software, the hardware, and the firmware would be made. The application area and design should be such that many of the operating system modules developed above can be used.

A project would then implement the system above the operating system level.

Finally, the demonstration of the DBMS may require the use of the DBMS by applications software. A project would support the integration of the system software and the applications software to complete the demonstration system.

Deliverables

(1) A working realtime database management system that demonstrates the viability of the interfaces.

(2) Information structure designs for data often required by DoD embedded computer systems that are not currently handled by database management systems, such as sensor and image data.

Cost and Benefits

The cost of this activity would be significant but modularly designed realtime DBMS could be reused in a number of DoD systems. Furthermore, this work would lay the groundwork for extending the DBMS to distributed systems for higher performance and reliability.

2.1.2.5 Interactions with Other Efforts.

Computer Security Center

DoD is operating a computer security evaluation center that is performing research in the architecture and construction of secure operating systems, DBMS, and transaction systems, among other things. The family of virtual machines developed by this subtask should be able to allow the development of secure versions.

MCF Operating System Project

Part of the work that is continuing on the Military Computer Family is the development of an Ada-based operating system.

Navy ALS/N Plan

The Navy is planning to design and build a realtime operating system as part of the ALS/N effort.

Applications Specific Area:

This subtask will make use of the research performed by the application-specific task on application program generators. This task will support the applications task by providing basic operating system functions to support the applications packages. There should be exchanges of information between the systems software subtask and the application-specific area to define the interfaces between applications software (i.e., the Ada packages) and system software in ways that also promote software/hardware synergy.

2.1.3 Software-Hardware Synergy Subtask

Software is one part of an entire system that also includes hardware. New hardware technologies and fabrication procedures admit the possibility of enhanced performance by realizing in hardware some of the system functionality that has traditionally been realized in software. This subtask focuses on the issues surrounding this

opportunity--in particular, tools for making software-hardware tradeoff decisions and methods supporting the co-design and co-evolution of hardware and software.

Exploiting the full potential of software-hardware synergy is not a simple matter of tradeoff analysis on where to place each function, but rather it opens the door to innovative architectures and interfaces which must be assessed as a whole.

The software-hardware synergy subtask would have two goals: to identify or develop methods for exploiting the potential of software-hardware synergy, and to develop and demonstrate tools oriented around these methods.

Tool and method development activities have been on-going for some time--for example, the VHSIC program has resulted in several hardware description languages that are of potential benefit in the co-design of hardware and software. In addition, tool and method development and evaluation receive concentrated attention in other initiative task areas, particularly the Support Systems Area.

The initial focus of the activities in the software-hardware synergy subtask would be to identify tools produced elsewhere that can be beneficially used for software-hardware tradeoff assessment, and methods (also produced elsewhere) that provide for the co-design of software and hardware. Once identified, these tools and methods might be demonstrated on application-specific projects in the application areas chosen in the application-specific area for concentrated attention.

The initial identification of potentially beneficial tools and methods would spawn another stream of activities which focus on developing improved tools and methods. This development would primarily be driven by the desire to provide complete and coherent co-design and co-evaluation methods.

The tool and method development work would account for the special needs of software-hardware co-design and result in demonstrably effective prototypes. These prototypes would be installed in the support environments produced by the Support Systems Area in order to perform evaluations and demonstrations. But the complete integration of production versions is not within the scope of this subtask. Rather, this is the responsibility of the environmental concerns subtask and the Support Systems area.

2.1.3.1 Analyze Assessment Techniques.

It is not likely that any particular software-hardware tradeoff assessment technique can be applied to all architectures and applications. For example, a particular technique may be intended to produce highly reliable systems without regard for size or performance. This technique would not be suitable for a smart munitions system, where size and weight are critical. Similarly, an assessment technique may provide a very good analysis of the correctness of an asynchronous system, which is appropriate for distributed architecture design, but not for the design of synchronous systems.

The purpose of this activity would be to match the assessment techniques with architectures and applications. One result of this activity would be a set of guidelines for selecting an assessment method that is appropriate for a particular application, architecture, and design method. An initial version of the guidelines based on existing techniques should be developed early. These could then be refined and elaborated to reflect new techniques developed in the future.

A side-effect of this activity would be an identification of techniques, and their supporting tools and compatible methods, for demonstration on trial projects. Another side-effect is an identification of new techniques, and their associated tools and methods,

that should be developed. This activity would therefore lead to the other activities within this subtask.

Description

This activity would start with a short project that evaluates existing assessment techniques. In this evaluation, techniques useful for the assessment of software-hardware tradeoffs would be identified and a framework for their classification developed. This framework would attempt to account for the system properties that can be identified as important.

After this initial accounting of what is available, two parallel projects would be started. One would match the identified techniques to particular application areas -- this would in essence be an elaboration of the classification framework to account for application characteristics in addition to system characteristics. The other would match the identified techniques to particular architectures by augmenting the classification framework with architecture characteristics.

Following these two projects, there could be two tightly linked follow-on projects. In the first, an initial set of guidelines would be developed for choosing an assessment technique (or a set of possible techniques) given a particular system characteristic, a particular application, and a particular architecture, and then these initial guidelines would be continuously honed and updated to reflect new techniques. In the second, the framework itself would be continuously honed and updated to reflect new techniques.

Deliverables

The assessment technique selection guidelines are the primary deliverable of this this activity. These guidelines will be available for stand-alone use, but they are also a primary input to the activity in which new techniques are developed.

Another deliverable is an identification of existing techniques for potential demonstration on trial projects.

2.1.3.2 Develop Co-design Methods. The assessment of existing techniques would uncover gaps, and this activity stream has the purpose of filling those gaps. In addition, it has the purpose of developing the tools supporting the techniques and the methods compatible with them. Thus, the emphasis is on developing new techniques and the driving force would be both holes in the set of assessment techniques and consideration of how the techniques are supported by tools and fit together to provide coherent software-hardware co-design methods.

This would be the primary activity for this Software-Hardware Synergy subtask. It would be initiated by the preliminary evaluation of existing techniques in order to make it account for and complement on-going work. It would produce the primary deliverable of this subtask -- tools for software-hardware tradeoff assessment and methods for software-hardware co-design and co-evolution. These tools and methods would then be integrated, by Support System Area activities, into the support environments delivered by the initiative to practitioners after the value of the tools and methods was demonstrated by the activities discussed in the section on the Environmental Concerns subtask.

Description

The primary activity would be an effort directed at developing complete and coherent software-hardware co-design methods that incorporate software-hardware tradeoff assessment techniques and are supported by tools implementing these techniques. It is difficult, at this point, to provide details for this activity, but a reasonable sequence of events would seem to be:

(1) Investigate the applicability of existing methods; this would utilize the results of the methodology study performed in the Support Systems Area.

(2) Identify one or more existing methods of highest benefit for the co-design of hardware and software and use it in some simple, trial development efforts.

(3) Develop new methods by enhancement and extension as warranted.

Tightly coordinated with this central activity would be two parallel efforts which serve to support it. First, there would be the project directed at providing the necessary tradeoff assessment techniques. Many of these could be obtained from outside activities, having been identified in the previously discussed assessment technique analysis activity. Others would have to be developed.

The other parallel project would be directed at developing the tools necessary for supporting the co-design methods. For the most part, these tools would implement the techniques for software-hardware tradeoff assessment. In addition, however, tools may have to be developed to provide support directly to the use of the methods. This project would be coordinated with the similar activities in the Support Systems Area.

Deliverables

The products here would be the co-design and co-evolution methods and their supporting tools. These would be delivered to the demonstration activity (see next subsection) for evaluation and subsequent delivery for integration into the STARS support environments.

2.1.3.3 Demonstrate Methods and Tools. The previously discussed activities would produce several methods, along with supporting tools, for the co-design of hardware and software. This activity is intended to demonstrate the utility of these methods through their use in actual development projects. The methods are demonstrated by

using them to design prototype systems selected from the areas recommended by the Application Specific Area. Although the users of the methods may be distinct from the developers, they should be provided with incentives to use the design methods effectively and to support the collection of data for the evaluation of the design system. During the process of these extended evaluations, design decisions should be recorded so that the design process can be captured and later analyzed. A result of this activity would be demonstrated methods that can be incorporated into the environment fielded by the Support Systems Area.

Description

The first project would be to select the applications to be addressed. These applications should be chosen from the areas recommended by the Applications Specific Area's study of potential areas for demonstration. They should also account for current or upcoming DoD projects so that there will be development efforts using traditional methods going on in parallel.

The next project is to design a data collection procedure for the pertinent data needed for later evaluations. Automated tools may have to be implemented to support this data collection.

The actual trial development efforts should require about two years. The initial 6 months might be used for installation of the supporting tools into the then-current version of the support environment so that it may be used. This might be followed first by a 6-month design phase, and then by a 9-month implementation phase. The final three months might then be needed to prepare for final demonstration of the developed products.

After demonstration, the data collected during development and demonstration are analyzed and the new methods are compared with traditional methods.

Deliverables

The direct result of this activity would be an evaluation of the methods. In addition, the tools supporting the methods, while primarily delivered toward the end of the demonstration period, could be delivered earlier if preliminary evaluation indicates substantial value.

2.1.3.4 Interactions with Other Areas.

DoD VHSIC Program

The VHSIC program has, as part of its Phase III projects, the development of a VHSIC hardware description language. This subtask would build on the VHSIC effort by evaluating hardware description languages developed by VHSIC.

Measurement Area

This subtask must be closely coordinated with the measurement task, since it will be developing the measures and experimental techniques helpful in the demonstration and evaluation of methods and tools.

Support Systems Area

This subtask would also be related to the support systems area, since maintenance and measurement of the description of a software-hardware interface requires a formal definition that can be manipulated by a computerized support system. The concepts for doing rapid prototyping that are developed by the support systems area should be examined for applicability to software-hardware systems.

Extensive work has been done on languages for specification of systems [Zave 82, Riddle et al. 78, Bell et al. 77, Ross 77, Teichrow and Hersey 77, Estrin 78, Penedo et al. 81]. Some of these specification languages are appropriate for the design of systems including both software and hardware. The key task at this point is using

these systems, evaluating their effectiveness for real design problems, and integrating these tools into a system for the design of systems.

Application Specific Area

In order to focus the research and promote the transfer of technology, the software-hardware synergy subtask should develop software using the methods and tools to do prototype designs. The prototype designs would be chosen in the applications areas identified by the applications-specific task.

Acquisition Area

The assessment tools developed by this task would be available for use by the acquisitions task. Since a major factor in the cost-effectiveness of a system is the nonrecoverable engineering design cost, the kinds of make-buy decisions that are a critical part of the acquisition process are also critical factors in the design. Thus, good tools for acquisitions are part of the set of tools needed for effective design and vice versa.

Management Area

Part of the results of this task would be information on how much effort to invest in system design, and what milestones need to be established to mark the completion of stages of a system design. Specification languages and design assessment procedures developed by this task will provide a means of communication between people doing the software design and people doing the hardware design.

Reliability Subtask

The techniques developed by the reliability subtask to reduce system errors will be included in the design tools.

Many embedded computer applications require highly reliable systems. The design of reliable systems is an important area for achieving software-hardware synergy. For example, software-hardware synergy is necessary in order to implement flexible (i.e., software controlled) recovery and (hardware) reconfiguration strategies [Hopkins et al., Wensley et al.]. This means that reliability specifications must be refined in parallel with the refinement of the system design, and that models for the effectiveness of software controlled recovery from hardware faults must be developed. The reliability subtask will develop the models, and this subtask will use them and evaluate them in the context of system design.

2.1.4 Reliability Subtask

The goals of the reliability subtask are methods and tools for producing highly reliable systems. The methods would be developed for ensuring the quality of all components of systems (including software, hardware, and documentation) throughout their life cycle, for removing faults from systems, and for designing systems that can tolerate faults. The tools would aid the system designer, implementor, or maintainer in producing highly reliable systems. They would be derived from the methods, and would be integrated into the support environment.

The reliability subtask would emphasize fault prevention through fault avoidance and fault detection techniques, and system fault tolerance. This subtask would emphasize quality assurance techniques for all stages of the life cycle, and would evaluate the costs and benefits of doing proofs of correctness for important system properties such as security and for algorithms developed during system design. Its use for critical pieces of code should also be evaluated. This subtask would examine techniques that support the earliest possible fault detection and removal.

The reliability subtask would evaluate fault tolerance techniques for systems, emphasizing techniques that employ software-hardware synergy to improve system reliability. This work will build on the research in fault tolerant systems being conducted by NASA.

2.1.4.1 Program Testing and Analysis. The purpose of this activity is to develop program testing and static analysis tools and methods. There are a variety of practical methods and tools for generating test data for programs, analyzing programs for correctness properties, and maintenance of program testing and analysis data. These tools and methods need to be integrated and brought into production usage.

Description

The first project is to develop and use production quality test data generation and coverage tools for practical methods such as branch testing, data flow testing, weak mutation analysis, and mutation testing.

The second project is to develop systematic guidelines for functional testing of programs. It would develop test generation methods which are based on information in requirements, specifications, and design documentation. This project would include a systematic elaboration of functional testing through checklists for functions or other tools.

The third project will develop and use production quality static analysis tools involving both traditional static analysis and general approaches to static analysis for concurrent and distributed systems.

Deliverables

(1) Program testing and static analysis tools.

(2) Section of verification and validation guidebook on program testing and static analysis.

2.1.4.2 Develop Methods for Fault-Tolerant Systems. The purpose of this activity would be to develop methods and tools for designing and programming of fault-tolerant systems. The emphasis of this activity is on achieving tolerance to both software and hardware faults through a synergistic combination of hardware and software components.

Description of Projects

The first project would be an evaluation of current recovery mechanisms and the classes of faults/errors for which they are effective. These recovery mechanisms and the classes of faults which they cover will be compared with the classes of errors and faults that have occurred in production embedded computer systems, particularly realtime systems. This comparison would determine for which kinds of faults/errors the systems should have been made fault-tolerant. A framework for classifying and understanding fault-tolerant issues and techniques should be identified or developed. The second project would determine how effective currently proposed fault tolerance methods and programming language features are for use in a uniform systematic approach to fault tolerance. The the third project would be to prepare a handbook for defining the requirements of, for designing, and for implementing highly reliable systems through the use of fault tolerance.

Deliverables

(1) A study of the potential effectiveness of proposed fault-tolerance techniques for faults/errors occurring in real systems.

(2) A framework for fault-tolerance

(3) Techniques and automated tools for designing and implementing fault tolerant systems.

(4) Fault tolerance handbook.

2.1.4.3 Develop Prototyping as a Reliability Tool. Prototyping can be thought of as a requirements validation tool. It could make it possible to compare proposed system configurations and designs with the informal expectations of the user.

Description of Projects

The first project would be to select requirements and specification techniques for evaluation. The techniques selected would be evaluated in terms of their support for rapid prototyping as an effective analysis approach. The selected techniques would be evaluated by experimental analysis. The second project would develop tools to support requirements and specification generation and analysis. The third project would develop the prototyping component of a requirements and specification generation and analysis system. This project would be carried out in cooperation with the support systems task. The fourth project would evaluate the effectiveness of prototype-based analysis for quality assurance during maintenance.

Deliverables

(1) Tools and methods for requirements and specifications development.

(2) Tools for prototype-based analysis of requirements and specifications.

2.1.4.4 Early Life Cycle Software Quality Assurance. The purpose of this activity is to identify the software quality assurance techniques that can be carried out early in the software life cycle and are critical to the development of high quality software. Since faults are much less expensive to remove if found early, there is also potential for cost avoidance.

Description of Projects

The first project is to identify practical and widely acceptable requirements, specifications and design methods for which it is possible to develop systematic quality assurance or verification and validation methodology. Methods and representations under consideration or selected by the Support Systems area or other parts of STARS should be emphasized.

The second project would develop specifications and design analysis techniques. This would include constructing tools for static analysis of early life cycle products.

Deliverables

(1) Specifications and requirements analysis tools.

(2) Design analysis tools.

(3) Section of verification and validation guidebook dedicated to early life cycle software quality assurance.

2.1.4.5 Proofs and Program Transformations. The purpose of this activity would be to develop guidelines for the use of proof of correctness and program transformations. Proofs can be used for proving the logical correctness of algorithms or for proving the consistency of programs with computational and performance models. They can also be used constructively as part of the program development process.

Description of Projects

The first project will develop guidelines for the use of proofs in proving the logical correctness of algorithms that are used in production systems.

The second project will evaluate security models and correctness proofs by experimental use of them for verifying critical properties of critical software modules.

The third project will evaluate the use of constructive proof techniques and current transformation technology for the development of correct production programs.

Deliverables

A guidebook on the use and effectiveness of proof and transformation technology and input to STARS planning concerning these techniques.

2.1.4.6 Interactions with Other Efforts.

Measurement Area

Cost-benefit analysis of formal proofs. Cost-benefit analysis of testing strategies. Impact of reliability techniques on life cycle costs.

Support Systems Task

Measurement of the impact of the use of support systems on the reliability of the software that they help generate. Automated procedures for software quality assurance and test generation would be developed and integrated into the support environment by this subtask to support the methods derived by the support systems area.

Applications Area

The applications task should support this subtask by identifying particular applications that require highly reliable software, in particular software modules that are suitable for correctness proofs. This subtask would support the efforts of the applications task to construct highly reliable software packages by developing correctness proofs for some packages as a proof-of-concept task. The software-hardware synergy subtask would support this subtask by developing prototypes of highly reliable systems that can be used as test cases for reliability experiments.

Management Area

The assurance of reliability of a system requires an ongoing effort. Several techniques for assuring the reliability of software depend on performing tests or reviews of the project at different stages in its development. These reviews are milestones that need to be organized and scheduled by the project management. The validation and verification handbook would support the management task in planning for reduced life-cycle costs. In order to provide this support, the reliability subtask must emphasize life-cycle oriented validation and verification.

Acquisition Area

The validation and verification handbooks generated by this subtask will support the acquisition task by including acceptance testing procedures.

Computer Security Center and Consortium

The computer security efforts include formal methods for modeling, specifying, and verifying as well as other security issues that may also relate to reliability. Of particular interest is the issue of when these methods will be suitable for transfer into an Ada-related tool set and environment.

2.1.5 Environmental Concerns Subtask

One environmental concern is the transition of tools and methods to the STARS support environment. As with many of the initiative task areas, the Systems area includes activities that would result in tools and methods to support and guide practitioners. Ultimately, these tools and methods would be delivered to practitioners via the support environments fielded as part of the Support Systems Area's activities. Prior to this they must be developed, at least in prototype form, and shown to be of demonstrable value. The other subtasks

in this plan for the Systems area develop and demonstrate tools and methods. This subtask interfaces between this internal tool and method work and the work within the Support Systems area.

The other environmental concern is to permit the early use of Ada for important DoD target machines. This may mean that Ada produced code and existing code must work together on some targets. Access must be provided through Ada to systems capability already existing on the target.

Ada facilitates this approach through its separation of interface and implementation therefore, a mixed language environment may be used for major upgrades and new developments. As a practical matter the systems area does not address the maintenance of existing systems.

The systems area is not limited to Ada and may go beyond Ada as required by more advanced architectures. Note that the techniques that allow Ada to be used with older less modern languages may also be used in the future to work with new advanced ones.

A target system and support system are closely related during the useful life of a system. Although they may be separate they are coupled for support purposes. They may be separate, even if they have the same underlying base, because of differences between the configuration of the target and support systems. The continuing advance of computing technology will result in some target systems having sufficient capacity to host a support system. However, as the capacity of target systems increases it provides the opportunity for increased functionality. This will require more advanced support systems. As a result the separate and integrated relationship of support and target systems is likely to continue. Improving the quality of the support/target interface is an important part of the

overall system from the point of view of having to meet continuing mission requirements.

The first environmental concern would primarily be met by collecting together the various tools and methods developed and demonstrated in the other Systems subtasks, integrating them together on the basis of systems concerns, and delivering them to the Support Systems area for final integration into the support environments.

Supporting this, there are two activities. In the first, tool and method work outside this area is monitored with the dual purpose of easing their subsequent integration into the support environments and assuring that the tool and method work internal to this area is consonant with similar work done elsewhere. The other activity focuses on the development of tools and methods that are supportive over the entire range of concerns within the Systems area.

Thus, these activities focus on three different aspects of integration: the development of "higher-level" tools and methods that account for several concerns, the development of tools and methods that are consistent and compatible with those developed elsewhere, and the integration steps needed prior to final integration into the support environments.

2.1.5.1 Tools and Methods for Complex Systems. A number of current, research-level environment development projects have focused on tools and methods supporting the development of complex systems: multiprocessor systems, distributed systems, communications systems, etc. The tools and methods provided by the efforts are oriented to the concerns within the Systems a although most all of them focus purely on system functionality. As a group, therefore, they can be used to provide a set of tools and methods that account for a spectrum of properties and concerns spanning the Systems area.

This activity focuses on integrating the previously developed systems-oriented tools and methods, demonstrating their value and extending/enhancing them as warranted.

Description

The first project, lasting about a year, would be intended to consolidate and integrate the existing techniques. Most all of them are based on the concept of communicating sequential processes and attempting to integrate them is therefore feasible with relatively little work. The value of trying to integrate them would be first to understand their similarities and differences and second to understand their strengths and weaknesses as a group.

Following this, and lasting somewhat longer, would be a project intended to demonstrate the value of these techniques when faced with system level issues. Some of the previously investigated tools and methods, or some amalgamation of them, would be used in a variety of simple, trial systems development exercises. Many of these exercises would be natural outgrowths of the work in other subtasks. Some would be peculiar to the problems of developing tools and methods over a variety of systems related problems.

These demonstrations would serve to identify tools and methods that should be integrated into the support environments being prepared in the Support Systems area. They would also serve to identify directions for continued development of these tools and methods, an effort that would begin at the end of the demonstration project.

Deliverables

The tools and methods developed under this activity are delivered to the parallel integration activity for eventual delivery to the Support Systems area.

2.1.5.2 Monitoring. This activity tracks external tools and methods work and assures that the internal work is not duplicative but is compatible.

Description

This is a single, continuous activity that interfaces the internal and external tool and method work.

Deliverables

There are no deliverables from this task except that it has the responsibility for importing externally prepared tools and methods for internal use.

2.1.5.3 Integrate and Deliver. The various Systems area subtasks should produce a number of tools and methods. This activity would assure that these are well integrated, with respect to systems concerns, before they are delivered to the Support Systems area for full integration into the automated support environments. The intent is to provide a coherent set of systems-oriented tools and methods to the Support Systems area.

Description

This activity concerns the first level of integration of tools and methods developed in the Systems area. The various tools and methods will have been developed with different intents and goals. It is through this activity that they are integrated with respect to concerns such as information structures and command languages.

Deliverables

Sets of integrated systems-oriented tools are delivered to the Support Systems area.

2.1.5.4 Make Ada Usable in Existing Targets. In order to use Ada soon in these existing target environments where significant

interest and potential exist, a number of capabilities would be required or desirable.

- o Ada cross compilers from support system to target system
- o Ada interface to existing system software
  - operating system
  - database management systems
- o Ada interface to existing system hardware
  - common hardware
  - special hardware
- o Ada interface to special device programming in other languages
- o Ada cross development facilities so that target system and support system can be viewed as an integrated environment.

For some targets some of these capabilities are already planned outside the STARS program.

These capabilities would not only allow the use of Ada, but would also allow it to be incrementally used as changes or additions are made to existing systems, particularly for major upgrades. Potentially, this may provide a sensible migration path to Ada for some systems.

2.1.5.5 Interactions with Other Efforts. This subtask would need to interact with all the other subtasks in Systems and with Support Systems. In addition, it should monitor and interact with outside efforts such as:

- o AJPO for Ada developments,
- o NASA for reliability and fault-tolerance developments,

- o DoD Computer Security Center for computer security and formal verification developments,
- o DoD VHSIC for VHSIC tools and methods
- o DARPA VLSIC for VLSIC tools and methods
- o DARPA Super Computer for tools and methods related to advanced architectures
- o Service efforts
- o European and NATO efforts
- o Japanese efforts.

2.1.6 Focusing R&D on Future Mission Needs

The STARS Systems area strategy calls for analyzing planned future mission-critical systems to derive the needed systems and software property combinations required, and then using this to focus R&D. This activity begins this process.

The mission community is both a source of requirements and a target for the production quality results. The role of the research community is to find solutions to problems derived from the mission requirements. This derivation is a responsibility of STARS management and may be done in a variety of ways as a form of mission needs assessment. This would be a natural complement to technology assessment that is envisioned for the Software Engineering Institute.

Solutions from the research community will eventually be used in products that are acquired. Both research and product community resources are scarce. In addition to focusing on selected problems it is highly desirable that the results be as generic as possible so that they may be widely used over a range of missions with varying property value ranges, architectures, and sizes. Such generic results are said to be scalable.

Scalability is an important characterization of solutions found by the research community and products developed by the product community. The research community does not have the capacity or inclination to produce full scale production quality results. Therefore, it must find solutions whose prototypes can be scaled up. Only then can assurance be gained from prototype demonstrations and experimental verification. Finding solutions to hard problems and finding solutions that scale are both research problems because of the risks and uncertainty involved.

The product community does not have the capability to produce a large number of different products economically. It must find ways to exploit reusable packages to benefit from the economics of larger scale production.

The user community does not have the capacity to make major system changes if they are too expensive. Therefore it needs adaptable systems that gracefully grow to meet changing needs.

The notion of scalable results addresses all of these concerns.

The number of technologies and properties involved in the systems area generates a problem space that is enormous in both size and complexity. An effective search for solution must be guided by mission needs and knowledge of the space. Proposed solutions must be validated by theoretical and experimental evidence.

Integrated solutions should be sought rather than special cases whenever possible. The development of integrated results can also provide significant synergistic improvement.

STARS also needs to develop the means to deal with the research community to obtain scalable results. This might be done through multiple competitive research, design, prototype efforts or by other means and incentives. Certainly included are evaluation methods and

criteria that allow efforts to know what to aim for and assess if the criteria have been met.

Description

- o Analyze and consolidate mission systems needs
  - Analyze DoD mission development and upgrade plans and assess needs in systems area
  - Consolidate across future needs to provide systematic description of future needs in terms of property value combinations and time needed
  - Translate these system needs into research problems
- o Consolidate research focus
  - Identify an initial set of mission system problems and evaluation criteria
  - Apply and possibly extend existing or recent results to mission systems problems
  - Demonstrate solutions as experimental prototypes
  - Conduct R&D aimed at out-year targets and problems identified from mission needs. Consolidation of mission system needs and research focus may partially proceed in parallel. The results of needs assessment will become the research focus as they becomes better understood. The research results during early consolidation will be for a general set of problems. Early assessments should help identify research directions for later consolidation and for enhancement.

2.2 Enhancement Phase

The Enhancement Phase of the STARS Systems Area is described in terms of what kinds of activities are needed and why, but not specific plans for any activities. Specific plans may be derived from more detailed descriptions of activities as they are needed during STARS implementation. Activities for enhancement are more research oriented and more focused on mission needs then they those

in consolidation. The scope, objective, and a summary of activity are given.

2.2.1 Strategy

The scope of the STARS System Area Enhancement Phase should continue to emphasize major upgrades and new developments. Ada with an advanced environment produced by the STARS Support Systems Area combined with a consolidated systems technology base should be used as a vehicle for further improving the state of practice. Cooperation with activities outside STARS during consolidation should begin to produce results that can be incorporated. An aggressive systems research program focused on mission needs should be emphasized in enhancement.

The objective of the STARS Systems Area Enhancement Phase are to develop an integrated approach to producing higher quality systems of increased functionality on shorter schedules.

- o Advance the use of modern system engineering
- o Advance mission systems needs assessment
- o Advance research focus on needs with demonstration of new results
- o Exploit identified technology insertion opportunities for new results.

The strategy that will be used and the kinds of activities that should be performed will prepare the system community for the transition phase. The strategy for the STARS System Area Enhancement Phase continues to advance the state of practice and begins to advance the state of the art in response to mission needs. Enhancement will emphasize advancing the state of the art in response to mission needs. Continued advance of the state of practice will be achieved by producing results in a form that can be used through Ada with its enhanced environment. Ada should continue to be used as a vehicle

for encouraging advanced software and systems engineering practice. Broader system issues should be addressed and basic research continue to find architectures that can effectively achieve combinations of system properties at levels sufficient to satisfy mission needs. The enormous size and complexity of the system space that needs to be explored requires focused effort and experimental validation of proposed solutions.

Continued evolutionary improvements can be expected from efforts external to STARS partially as a result of STARS consolidation activities and criteria setting. The STARS Systems Area should aggressively pursue mission assessment policies that identify quantifiable mission systems needs (with evaluation criteria) and technology insertion opportunities. Aggressive research programs should focus on these needs and demonstrate the effectiveness of proposed solutions through experimental prototypes. Aggressive preparation for technology insertion should engineer successful research results into production-quality tools. Production-quality results available for insertion, continuing advances in higher level systems languages, and enhanced support environments should accelerate the advancing state of the art and reduce the lag in the state of practice. The result of enhancement will be the basis for transition based on the institutionalized strategy and process STARS will leave in place.

2.2.2 Activity Summary

The activities identified for the STARS System Area Enhancement Phase are stated in terms of what results are needed, but not specifically how they will be achieved. Some mild constraints may be applied to facilitate reuse of the results. Many of these activities would be more research oriented than many of the consolidation activities.

The activities are summarized as follows:

- o Enhance Mission Systems Assessment
  - Mission Systems Assessment during consolidation should have identified needs and communities of interest
  - Define properties of systems and way to quantify
  - Define mission systems problems sets
- o Enhance Mission System Research Focus
  - Basic research on systems properties and ways to effectively achieve combinations of properties toward an integrated systems approach
  - Start by combining a small number of properties and expand as understanding and need develop
- o Enhance Scale of Experimentation Experiment on methods for designing, specifying, and implementing systems that have properties of high performance, adaptability, reliability and others as needed
- o Enhance Preparation for Technology Insertion
  As experimental results emerge demonstrate that they can be production engineered and inserted
- o Software/Hardware Synergy
  Architecture for systems should include methods and tools for Software/Hardware Synergy. In addition to the use of conventional system structures, consideration should be given to alternative structures that exploit reduced hardware costs resulting from VHSIC and VLSI. The ability to tune a system for performance should be a criteria for assessing modularity. Results should provide basis for an integrated software/hardware systems development environment with access to VHSIC and VLSI rapid fabrication.
- o Environmental Concerns
  A more integrated approach to system integration and configuration is needed. An enhanced mission system interface to the support system environment and facilities to instrument the mission system during real operations are needed.

- o Nontraditional Architectures
  Make mission needs known and provide opportunities for nontraditional architecture to be demonstrated on mission problems and prepared for insertion as they emerge.

2.3 Transition

The Transition Phase of the STARS Systems Area will not be described in specific terms. Among the expected results of the enhancement phase are some techniques for improving the ability of systems technology to keep pace with mission systems needs. In addition, there will be some evidence that will provide insight on their effectiveness. During transition the experience gained should be used to institutionalize this "system" for producing continually enhanced systems. After DoD has researched the STARS goals, it will still need to pursue new frontiers. The legacy of STARS must be a DoD systems and software community, both researchers and practitioners, that effectively continues to meet DoD's constantly increasing operational mission needs.

3.0 REFERENCES AND SUPPORT MATERIAL

Many of the issues in the Systems area were also covered in Strategy for a DoD Software Initiative, Volume II (ODUSD(R&AT) 1 October 1982), which contains relevant appendices and bibliographies on distributed systems, database management systems, hardware-software synergy, and reliability.

3.1 Computer Architecture

(1) Special Issue on Parallel Processing, Computer 15, 1, Jan. 1982.

(2) Special Issue on Distributed Processing, Computer 11, 1, Jan. 1978.

(3) Special Issue on Distributed Processing Systems, IEEE Transactions on Computers C-29, Dec. 1980.

(4) ACM Conference on Functional Programming and Computer Architectures, Portsmouth, N.H., Oct. 1981.

(5) International Conferences on Parallel Processing, sponsored by IEEE.

(6) SPIE Symposia on Real Time Signal Processing.

(7) Annual Symposia on Computer Architecture, sponsored by ACM and IEEE.

(8) Siewiorek, D.P., C. G. Bell, A. Newell, Computer Structures: Principles and Examples, McGraw-Hill, New York, 1982.

3.2 Software-Hardware Synergy

(1) Alford, M.W. "a requirements engineering methodology for real-time processing requirements," IEEE Transactions on Software Engineering, SE-3, No. 1.

(2) Bell, T.E., D.C. Bixler, M. Dyer. "An extendible approach to computer-aided software requirements engineering," IEEE Transactions on Software Engineering, SE-3, 1, pp. 49-59, Jan 1977.

(3) Mead, C. and L. Conway. Introduction to VLSI Systems, Addison-Wesley, Reading, Mass. 1980.

(4) Estrin, G. "A methodology for design of digital systems - supported by SARA at the age of one," AFIPS Conference Proceedings, vol. 47 (1978)

(5) Hamilton, M. and S. Zeldin. "Higher Order Software -- A methodology for defining software," IEEE Transactions on Software Engineering, SE-2, 1, pp. 9-32, July 1976.

(6) Hopkins, A.L., T.B. Smith III, H.L. Jaynaran. "FTMP: a highly reliable fault-tolerant multiprocessor for aircraft," Proceedings of IEEE, vol. 66, no. 10, Oct. 1978, pp. 1221-1239.

(7) Penedo, M.H., D.M. Berry, G. Estrin. "An algorithm to support code-skeleton generation for concurrent systems," Proceedings of the 5th International Conference on Software Engineering, pp. 125-135, 1981.

(8) Riddle, W.E., J.C. Wileden, J.H. Sayler, A.R. Segal, A.M. Stavely. "Behavior modeling during software design," IEEE Transactions on Software Engineering, SE-4, 4, pp. 283-292, July 1978.

(9) Ross, D.T. "Structured analysis for requirements definition," IEEE Transactions on Software Engineering, SE-3, 1, pp. 6-15, Jan 1977.

(10) Silverberg, B. A. "An overview of the SRI hierarchical development methodology," SRI Technical Report, CSL-116, 1980.

(11) Teichrow, D., E.A. Hershey III. "PSL/PSA: A computer-aided technique for structured documentation and analysis of information processing systems," IEEE Transactions on Software Engineering, SE-3, 1, pp. 41-48, Jan 1977.

(12) Wensley, J.H., L. Lamport, J. Goldberg, M.W. Green, K.N. Levitt, P.M. Melliar-Smith, R.E. Shostak, C.B. Weinstock. "SIFT: Design and analysis of a fault-tolerant computer for aircraft control," Proceedings of IEEE, vol. 66, no. 10, Oct. 1978, pp. 1240-1255.

(13)Zave, P. "An operational approach to requirements specification for embedded systems," IEEE Transactions on Software Engineering, SE-8, 3, pp. 250-269, May 1982.

(14)Proceedings of ACM Symposium on Architectural Support for Programming Languages and Operating Systems, March 1982, Palo Alto, California.

3.3 Reliability

(1) Anderson, T. and P.A. Lee, Fault Tolerance: Principles and Practice. London: Prentice-Hall International, 1981.

(2) "Production of reliable flight-crucial software," NASA conference publication 2222.

(3) Hopkins, A.L., T.B. Smith III, H.L. Jaynarayan. "FTMP: A highly reliable fault-tolerant multiprocessor for aircraft," Proceedings of IEEE, vol. 66, no. 10, Oct. 1978, pp. 1221-1239.

(4) Howden, W.E. "Applicability of Software Validation Techniques to Scientific Programs," ACM Transactions on Programming Languages and Systems, 2, 1980.

(5) Howden, W.E. "Life Cycle Software Validation," Computer 15, 1982.

(6) Littlewood, B. "Theories of software reliability: how good are they and how can they be improved?" IEEE Transactions on Software Engineering, SE-6, 5, Sept. 1980, pp.489-500.

(7) Myers, G.J., "A Controlled Experiment in Program Testing and Code Walkthrough/Inspections," Communications of the ACM, Sept. 1978.

(8) Musa, J.D., "The measurement and management of software reliability," Proceedings of the IEEE, vol. 68, no. 9, Sept. 1980, pp. 1131-1143.

(9) Redwine, S. T., "An Engineering Approach to Software Test Data Design," IEEE Transactions on Software Engineering, SE-9, 2, March 1983.

(10)Wensley, J.H., L. Lamport, J. Goldberg, M.W. Green, K.N. Levitt, P.M. Melliar-Smith, R.E. Shostak, C.B. Weinstock. "SIFT: Design and analysis of a fault-tolerant computer for

aircraft control," Proceedings of IEEE, vol. 66, no. 10, Oct. 1978, pp. 1240-1255.


DATE:
ILME