TECHNICAL MEMORANDUM

WL-TM-91-123

MAINTAINING AN OPERATIONAL FLIGHT PROGRAM (OFP)

Charles P. Satterthwaite
WL/AAAF-3
WPAFB, OH 45433-6543

Software Concepts Group
Avionics Logistics Branch
System Avionics Division

Avionics Directorate
Wright Laboratory
Wright-Patterson AFB OH 45433-6543

FOREWORD

This technical memorandum was prepared by Charles P. Satterthwaite. This report documents Mr. Satterthwaite's experience as an F-15 Central Computer Operational Flight Program Maintainer at Warner Robins Air Logistic Center. The report was developed under Work Unit 99930000 in the Software Concepts Group, Avionics Logistics Branch, System Avionics Division, Avionics Directorate, Wright-Patterson AFB OH 45433-6543.

The author wishes to thank his colleagues at Warner Robins' ALC/LFLEO and at Wright-Patterson's WL/AAAF.

This Technical Memorandum has been reviewed and approved.

Charles P. Satterthwaite

CHARLES P. SATTERTHWAITE
Electronics Engineer

Robert L. Harris

ROBERT L. HARRIS
Group Chief
Software Concepts Group

Donna M Morris

DONNA M. MORRIS
Branch Chief
Avionics Logistics Branch

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1a. REPORT SECURITY CLASSIFICATION
Unclassified

1b. RESTRICTIVE MARKINGS

2a. SECURITY CLASSIFICATION AUTHORITY

2b. DECLASSIFICATION / DOWNGRADING SCHEDULE

3. DISTRIBUTION / AVAILABILITY OF REPORT
Approved for public release;
Distribution unlimited

4. PERFORMING ORGANIZATION REPORT NUMBER(S)
WL-TM-91-123

5. MONITORING ORGANIZATION REPORT NUMBER(S)

6a. NAME OF PERFORMING ORGANIZATION
Wright Laboratory

6b. OFFICE SYMBOL (If applicable)
AAAF-3

7a. NAME OF MONITORING ORGANIZATION
WL/AAAF-3

6c. ADDRESS (City, State, and ZIP Code)
Wright-Patterson AFB OH 45433-6543

7b. ADDRESS (City, State, and ZIP Code)
Wright-Patterson AFB OH 45433-6543

8a. NAME OF FUNDING / SPONSORING ORGANIZATION
WL/ASD (AFSC) Avionics Directorate

8b. OFFICE SYMBOL (If applicable)

9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
In-house

8c. ADDRESS (City, State, and ZIP Code)
Wright-Patterson AFB OH 45433

10. SOURCE OF FUNDING NUMBERS

PROGRAM ELEMENT NO.	PROJECT NO.	TASK NO	WORK UNIT ACCESSION NO.

11. TITLE (Include Security Classification)
Maintaining An Operational Flight Program (OFP)

12. PERSONAL AUTHOR(S)
Charles P. Satterthwaite

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
Summary	FROM May 86 TO Oct 91	91 Oct 1	28

16. SUPPLEMENTARY NOTATION

17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	Operational Flight Programs Avionics Integrated Support Facility

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

This report documents the experiences of an F-15 Central Computer (CC) Operational Flight Program (OFP) maintainer while he worked in support of that weapon system at Warner Robins Air Logistic Center's F-15 Avionics Integrated Support Facility (AISF). The report addresses the role of an OFP; the mechanisms of changing OFPs; the testing, integration, and documentation of OFPs; and the training requirements of OFP maintenance environments. Included with the report is a glossary, which expands on OFP peculiar jargon.

20. DISTRIBUTION / AVAILABILITY OF ABSTRACT
☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT. ☐ DTIC USERS

21. ABSTRACT SECURITY CLASSIFICATION
Unclassified

22a. NAME OF RESPONSIBLE INDIVIDUAL	22b. TELEPHONE (Include Area Code)	22c. OFFICE SYMBOL
Charles P. Satterthwaite	513-255-3947	WL/AAAF-3

BIOGRAPHICAL SKETCH

CHARLES P. SATTERTHWAITE
WL/AAAF-3
WPAFB, OH 45433-6543

Charles P. Satterthwaite is an Electrical Engineer in the Software Concepts Group, Wright Laboratory, Wright-Patterson AFB, OH. Mr Satterthwaite has over five years of experience as a Government Engineer which include three years with the Air Force Logistics Command and over two years with Aeronautical System Division's Wright Laboratory. His engineering background includes F-15 Operational Flight Program Maintenance at Warner Robins Air Logistic Center and Embedded Computer System Software Research and Development at Wright Laboratory. His professional affiliation is with the IEEE. He received a B.E.E. degree (electrical engineering) from the University of Dayton and a B.S.N.R. degree (natural resources) from the University of Michigan.

MAINTAINING AN OPERATIONAL FLIGHT PROGRAM (OFP)

1 October 1991

Charles P. Satterthwaite/Engineer

Software Concepts Group (WL/AAAF-3)
Wright-Patterson AFB OH 45433-6543

TABLE OF CONTENTS

Table of Contents continued

1 October 1991

MAINTAINING AN OPERATIONAL FLIGHT PROGRAM (OFP)

1.0 INTRODUCTION

This paper gives an overview of how Operational Flight Programs (OFPs) are maintained and developed. The paper briefly describes an OFP; how an OFP works; what causes change in OFPs; how an OFP is changed; how an OFP is tested; how an OFP is documented; how to train OFP maintainers; and how to measure the performance of OFPs.

2.0 DISCLAIMER

This paper is written from the experience of an F-15 Central Computer Operational Flight Program Maintainer. It is assumed that the processes experienced by this maintainer overlap into other OFP Maintenance Environments.

3.0 OVERVIEW

3.1 OFP Defined

An Operational Flight Program is the software program of an embedded computer system which enables that system to perform its interactive tasks as designed.

3.2 OFP Examples

The F-15 CAK-1070 is actually a block cycle of two OFPs. CAK is the F-15 APG-63 96k Radar's OFP while 1070 is the F-15 Central Computer's OFP.

3.3 OFP's Role In The Weapon System

Embedded computers are increasingly called upon to provide high-tech solutions to complex multiple threat type environments for today's generation of weapon systems. The guts of an embedded computer is its software, which is the OFP. In understanding the role of an OFP, one must thoroughly understand the threat, the weapon system, the mission, and the embedded computer system.

3.4 Maintenance Of An OFP

OFPs are easier to maintain than mechanical units. An OFP does not break or wear out. It does not make a mistake, but it could contain faulty logic, which it would follow faithfully. To maintain an OFP, one must be able to manipulate its sub-functions, without destroying its known integrity. There also has to be a method by which the OFP can be interactively examined and tested. Maintenance of OFPs requires a library of current and historical files which encompass the various block cycles and their unique versions as well as documentation, specialized test patch software, and any other records related to the OFP.

4.0 HOW DOES AN OFP WORK?

The Operational Flight Program (OFP) literally is the software portion of a embedded computer system. The computer and its periphery interfaces make up the system hardware. The hardware enabled by the OFP software describes the whole system.

The embedded computer system has partitioned memory which is filled with some type of machine level (binary) code. The OFP is loaded into this partitioned memory and, when enabled, empowers the whole system to perform its desired functions. Each embedded computer system has an instruction set which is burned into its Read Only Memory (ROM). The instruction set allows the embedded computer maintainer access and the capability to optimize the remaining partitioned memory. The level of sophistication of a embedded computer system is described by its instruction set, its memory, and its throughput.

5.0 WHAT DRIVES A CHANGE TO AN OFP?

Given a working OFP in a working system, why would I ever want to make changes? One reason would be that the users of the system would require an altered mission. An example of this would be the Tactical Air Command (TAC) requesting an Engineering Computer Change. A typical TAC Form 37 would be a request to provide a clearer display for the pilot under some given condition. Another reason would be that some flaw is discovered while the embedded computer system is operational. Some combination of events might cause partial or total system failure, prompting a review and redesign in the affected areas of hardware, software, or both.

6.0 HOW IS AN OFP CHANGED?

6.1 Diagnosis/Analysis/Isolation/Integration/Test

Given the task of changing an OFP (making a new version or even a new block cycle), several steps are followed to bring about the change. First, the requested change(s) is/are diagnosed so that their purpose is understood. Engineers and pilots don't always view life in parallel, so careful review keeps the OFP maintainer on track. Once the OFP maintainer thoroughly understands the change request, he makes an analysis as to which OFP areas he must alter. Usually the OFP is made up of a series of modules with specialized functions which will be covered in more detail later. A typical TAC Form 37 change might impact three modules of a forty module OFP. The OFP maintainer will next isolate these modules by making copies of them and implementing his design changes to his copies. The OFP maintainer integrates his assembled modules by linking it together with the other unaltered modules to form his own unique OFP. The OFP maintainer's final task is to test out his OFP by putting it through an acceptance test procedure, which wrings out the new OFP. For a sizable OFP with significant TAC Form 37 change requests, several maintainers would follow these procedures simultaneously, and then a lead maintainer would integrate and test the new OFP.

6.2 The Target Processor

In order to make changes in OFPs, a thorough understanding of the target processor (the embedded computer system) is required. It is the software engineer's nightmare that most target processors have their own peculiar instruction sets. This is one explanation for the proliferation of compilers. In order to make an target processor function efficiently, the parameters of that processor must be understood. These parameters include the processor's instruction set (usually assembly), the processor's input/output, its throughput, and the inter-dependencies the processor has, such as periphery devices, interrupts, and shared memory.

6.3 The Modules/Functions Of The OFP

Many OFPs are made up of modules which partition the OFP into its functions and sub-functions. A typical Fire-Control Computer contains air-to-air, air-to-ground, navigation, control and display, executive, Heads-Up-Display (HUD), and over-load warning functions, each of which has one or more sub-functions. An example of a sub-functional module would be a air-to-air 50 cycles per second module. The air-to-air function might be made up of three modules (10/sec, 20/sec, and 50/sec). Many of the modules would have inter-dependencies. For example the executive modules would determine the timing and priority scheduling among the entire OFP.

6.4 The Weapon System/Mission

In order to make OFP changes, a maintainer must understand the weapon system for which his embedded computer is a part, and the mission for which that weapon system is required. Many times the availability for new functions in a embedded computer system are limited, so that a trade-off analysis must be performed in order to optimize the mission and the weapon system. A sub-function which is rarely or never utilized might be sacrificed in order to accommodate a new requirement of higher priority to TAC.

6.5 The Support Environment

In order to maintain an OFP, the maintainers require a dedicated computer system and a simulation environment. The dedicated computer system allows the maintainer to access OFPs as well as copy and alter OFPs as required. The simulation environment allows maintainers to run their OFPs enabling them to debug and test interactively.

The hardware of a dedicated computer system usually includes main-frame computers (or powerful engineering workstations), various types of printers, various disk storage devices, networking, and several access terminals. An example used by the F-15 Central Computer OFP Maintainers is the Harris Operating System with Harris 800 and 1200 Mainframes, as well as a complimentary host of Harris Printers, Disk Drives, and Reel to Reel Drives.

Some confusion can be cleared up here between an embedded computer and a dedicated computer. The embedded computer is the target processor which is part of the weapon system. A dedicated computer is outside the weapon system and is used to support the embedded computer system.

6.5.1 The Dedicated Computer System

The dedicated computer system allows maintainers to keep copies of all the OFP's software, documentation, peculiar support utilities, and flow diagrams. It is through the dedicated computer system that a maintainer gains access to OFPs; edits or creates new versions of OFPs; updates support documentation; manages the OFP configuration; trains; and enhances the support environment with additional hardware or software.

The software of the dedicated computer system is usually peculiar to the hardware vendor. The Harris System mentioned above has its own operating language, editors, compilers, linkers, etc. The OFP maintainers also develop specialized utilities (usually in FORTRAN) to expedite and enhance the procedures they follow in maintenance and development of OFPs.

The dedicated computer system is also comprised of system conventions which help to maintain configuration management, security regulations, and proper operation of the dedicated computer system.

6.5.2 **Simulation Environment**

OFPs must have a means by which to operate real-time, that is, loading them up in their target processor and exposing them to the range of conditions (or a reasonable sub-set of those conditions) they encounter when operational. This means should allow the maintainer to actively debug the OFP. The degree of complexity of the OFP's environment is directly related to the complexity of this simulation environment. In the case of a typical fire control computer, you need a means to represent the full-up avionics suite and the dynamic environment the fighter encounters. You also need an interface to all cockpit controls and switches and an interface between the dedicated computer system and the simulation environment. Finally you need competent maintainers who know how to make the system work.

The simulation can range from a fully operational weapon system (flight testing is very expensive) to an all-software engineering workstation. Usually the simulation is a representative set of the weapon system's Line Replaceable Units (LRUs) with software emulating the cockpit and the dynamic environment.

Interaction with the simulation environment is through the dedicated computer system. Simulation utilities hosted on the dedicated computer system allow you to load an OFP into its target processor and exercise it dynamically or statically. These utilities also allow you to record, patch, debug, freeze, and initialize the OFP.

6.6 **The Avionics Integrated Support Facility (AISF)**

The facility which houses the dedicated computer system(s) and the simulation environment(s) is the Avionics Integrated Support Facility (AISF). Another name for the AISF is the Centralized Software Support Activity (CSSA). The AISF supports one or more embedded computer systems and their OFPs.

6.7 **Making an OFP Change (A Run-through)**

A good way to understand how one would change an OFP is to run through an imaginary change process. George has been tasked by his management to respond to a TAC Form 37 (Computer Engineering Change Request) which requires the "Break X" on the Heads Up Display to be replaced by a "Break M" when the aircraft is in imminent danger of collision. "Break X" is a flashing display presented to a pilot when his aircraft is in danger of colliding with some other airborne vehicle or debris. "Break M" is an imaginary display similar to "Break X" and used for this

example. George verifies TAC's desires by contacting pilots who have requested this change, and further learns that "Break X" distorts visual clarity when the pilots are engaging enemy targets. Through a simple exercise, the pilots found a "Break M" would greatly enhance their ability to fight and fly. George does an analysis of the OFP and determines that changes must be made in four OFP modules in order to accommodate the requested change. He also determines that the changes will result in no additional lines of code added, since only fonts and variable values have to be changed. George makes copies of those files containing the affected modules, he makes his changes, he assembles his copies, and then he links them together into an OFP he calls OFP_GEORGE. George then Loads OFP_GEORGE into his facilities engineering workstation and validates his design through his facilities acceptance test procedure.

7.0 HOW IS AN OFP TESTED?

The ultimate test of an OFP is that it becomes the operational version. But several layers of testing exist before OFPs are accepted. Flight tests are expensive, as are full-up simulations. But some confidence can be gained through wringing the OFP out on its software simulated environment. The process which wrings an OFP out is called the acceptance test procedure (ATP), which will be discussed in more detail in Topic 7.4.

Various other tests are required in the software development life cycle of OFPs. These include tests of the target processor (and its environment), peculiar tests, and the Operational Test and Evaluation (OT&E).

7.1 The Requirement to Test

The requirement to test is related to the confidence desired of the targeted system or sub-system. Low level testing might be sufficient for minor operational adjustments such as flight-line data entry. But processes affecting life support, terrain following radar, and navigation (to name a few) require highly integrated testing. The ultimate test is a system operationally used repeatedly with all its performance monitored and recorded.

7.2 Testing Assumptions

The following assumptions are made before further discussing OFP testing.

- All the modules of the OFP are assembled and linked
- Security arrangements are in place
- The simulation environment will support the tests
- The OFP maintainers are experienced testers

7.3 Processor Peculiar Tests

7.3.1 Utilities Developed For Processor's Peculiarities

The OFP maintainer must be thoroughly familiar with his target processor and its unique features. These target processors might require specialized testing to understand their behavior and to optimize their performance. An example of a specialized test utility is the F-15 OFP Central Computer's BASEREG program which checks the target processors base registers for errors caused by deleting or adding code. Another example is the same system's PROBER program, which checks all paths of an OFP to make sure they are utilized or exercised.

7.3.2 Utilities Developed For Specialized Environmental Testing

The testing environment for OFPs might require specialized testing to insure proper functioning of an integrated system. Programmable Monitor And Control (PMAC), as an example, allows a maintainer to interact with his running OFP. Other specialized environment testing might indicate system or sub-system failure, offer a means to integrate multiples of OFPs (radar, central computer (CC), and electronic warfare (EW)), and allow patching of the OFP code for various environmental conditions such as threats, initial conditions, positions, etc.

7.4 The Acceptance Test

The OFP maintainers primary test is the acceptance test procedure (ATP). This test is designed to wring out an OFP to a degree that it can be released with confidence to flight test and then operational test and evaluation.

The ATP is a chronological check of the OFP's responses to inputs. Inputs include switch positioning, preset conditions such as altitude or airspeed, and hardware interrupts to name a few. The OFP loaded into its embedded computer and hosted on its simulation environment responds to these inputs in the form of static or dynamic displays, which can be checked against expected results.

7.5 What Is An Acceptable Level Of Testing?

The users of an OFP, along with the OFP's management, determine an acceptable level of performance for an OFP. This is a non-trivial task since a typical fighter OFP might have thousands of check-points required to ensure safe utilization.

7.6 The Baseline Acceptance Test Procedure

The baseline acceptance test procedure (ATP) is the ATP which complimented the most recent version of the OFP (the last block cycle change). An ATP should be developed concurrently with its OFP. That is to say any additions, deletions, or modifications to the OFP should be paralleled by the ATP. Typically, ATPs don't change drastically between OFP updates because the major functionality of the OFP system remains the same.

7.7 Static Tests and Dynamic Testing

Static tests are tests which are not time dependent. Given an input or a combination of inputs, there should be expected responses. As an example, in Gun Mode, I expect to see a Gun Reticle. The Gun Reticle is a circle displayed to a pilot on the Heads Up Display (HUD). Dynamic tests are much more complicated since they are time dependent. They might require a sequence of inputs over some interval or some feed-back tape in order to ensure proper functioning of some sub-set of the OFP. An example of a dynamic test is to observe an expected Signal-to-Noise Ratio (SNR) improvement as I decrease in range on a target that I am tracking with my radar.

7.8 Classified Tests

Arrangements must be made for some OFPs to do classified testing. This requires the facilities and maintainers to be cleared to the level of classification of testing. It also requires a means by which to properly store and maintain classified testing documentation. It is often convenient to isolate classified portions of OFP testing so that non-classified OFP testing can proceed unhindered.

7.9 Automated Tests

As the complexity of OFPs increases with software usage, the ability to manually perform acceptance test procedures (ATPs) decreases or the ability to fully test OFPs decreases. The F-15 Central Computer OFP Acceptance Test currently takes two man weeks. Much of this F-15 ATP is static testing in which the maintainer is flipping switches and verifying displays. This ATP time requirement for manual check-out will soon be man hour prohibitive for new versions of OFPs with orders of magnitude more code. One possible solution is to automate as much of the ATP as possible by utilizing the shared memory and remote control features of software engineering work stations. One possible means of implementing this automation is through a tool developed at Wright Laboratory called Automatic Validation (AUTO-VAL).

7.10 **Flight Tests**

When an OFP passes its acceptance test procedure, it might require a flight test(s) to further qualify it as wrung out. Some areas of OFPs can't be checked out through laboratory testing. Efforts are made to minimize Flight Testing because of the enormous costs involved ($5-10K an hour or more).

7.11 **Operational Test And Evaluation**

Operational test and evaluation is where the OFP must meet the approval of those pilots who will use it. These pilots (or users) have their own check-out procedures which can include live firing of munitions, lock-on and destruction of drones, and navigational exercises to mention a few. Obviously the slightest glitch in performance draws immediate fire on software maintainers.

7.12 **Iterative Nature Of Operational Flight Program Testing**

Usually OFPs are not acceptable in their first cut, even when they go through OT&E. Five or six cycles through the testing process is not unusual. Much of this is related to the complex nature of OFPs, poor pilot-to-engineer feed-back-loops, and changing mission requirements midstream in OFP development.

8.0 **HOW IS AN OFP DOCUMENTED?**

Several types of documentation exist to support the development and maintenance of OFPs. Technical Orders (TOs) are most prevalent with Version Description Documents (VDDs) being most common. Documentation such as the Technical Description Document (TDD), TAC Form 37 Engineering Change Requests also exist, plus a host of ancillary notes generated when development work occurs. Most documentation occurs after an OFP is wrung out. The lead maintainer writes a synopsis of changes made between versions, which gets interpreted into the VDD and the TDD. TOs usually follow several months after an OFP checks out.

Documentation is generated preceeding, concurring, and ending each OFP block cycle change or version change. Proper documentation allows each level of the OFP software development cycle to be visible and specified to the level of detail required.

8.1 The Requirement To Document

A limiting factor in maintaining Operational Flight Programs is documentation. Documentation is required at every phase of the software life cycle. Documentation defines requirements, specifies designs, directs testing efforts, and maintains configuration, to name a few. The Operational Flight Program maintainer uses documentation to pin-point his users requirements and to optimize the OFP to accommodate those changes. The maintainer is also responsible for generating and updating documentation which explains to various levels of OFP users the rationale and methods of software changes.

8.2 Levels Of Documentation

Several layers of OFP users and developers require documentation to perform their specific tasks in the OFP's life cycle. These include the maintainers/developers, management, pilots, and aircrews.

8.3 Documentation Useful To Maintainers/Developers

The maintainers and developers of Operational Flight Programs require the most detailed technical documentation in order to perform their mission. They utilize technical orders, Version Description Documents, Technical Description Documents, Detailed Design Reviews, Flow Diagrams, Operational Test and Evaluation Reports, and Detailed System Manuals. They also require up-to-date documentation for acceptance test procedures, instructions for the operating system and support system as well as system peculiar documents such as security, safety, and environmental.

8.4 Documentation For Management

The management of facilities which maintain and develop Operational Flight Programs are concerned about the resources required to perform their Mission. They are responsible for justifying the man-power, the real-estate and facilities, the program priority, and the fiscal obligations of the OFP maintenance activities. Most of these managers have extensive experience as OFP maintainers, so they understand the detailed documentation their maintainers and developers utilize. The documentation they require gives them project overview, a measure of performance, and estimates of cost. Much of the documentation these managers utilize comes from their own experiences, since they are the present day first generation software engineers who set up the existing OFP maintenance programs.

8.5 Documentation For Users (Pilots And Crews)

Once an Operational Flight Program is released to the field, it must be supported by documentation which explains its new features and how to use it. Pilots and aircrews are interested in the end product functionality of their weapon system. Technical Orders and Version Description Documents explain to these users the enhancements to their system and how to generate them.

These users continue the Operational Flight Program Software Life Cycle by issuing problem reports and by requesting new enhancements, which come back to OFP maintainers as engineering change request documentation.

8.6 Historical Documentation

Historical documentation is all the documentation that has been used to generate and maintain a Operational Flight Program. This documentation is necessary in order to maintain the integrity of the Operational Flight Program. It also allows maintainers to support older versions of OFPs or reference versions for possible reutilization of old features.

8.7 Ideal Documentation

Ideal documentation allows every individual in the Operational Flight Program Life Cycle perfect access to the required level of detail through the entire life of the OFP. The processes and variables of the OFP can be traced and cross-referenced from whatever level a maintainer is working.

8.8 Automated Documentation

The iterative nature of maintaining Operational Flight Programs tends to cause the documentation process to occur as a final step, rather than with each iteration. This causes much valuable information to be lost. The capture of mistakes is necessary because you know what not to do. Abandoned efforts might be called upon in future OFP change activities.

Unfortunately, documenting changes is tedious work so it is put off as a last phase effort. Most of the interim information gleaned in development is lost.

Documentation tools could be built into the maintainers toolbox so that whenever he assembled his source code, some minimum set of documentation would be recorded. This tool could capture the user, date and time, files altered, and prompt for explanations of additions, deletes, new variables, and logic flow.

9.0 HOW DO YOU TRAIN OFP MAINTAINERS?

The training of OFP maintainers requires multiple levels of instruction which include weapon system, target processor, dedicated computer system, simulation system, and integrated testing, plus the facility requirements such as security that the new maintainer must learn. Often the new man is on his own, without a proven method or mentor to bring him up to speed.

9.1 The Requirement To Train

The combined skills required to effectively maintain OFPs depends on an adaptable environment where continuous training provides the impetus of change. Oftentimes, no one individual can handle the workload which an OFP demands. Unfortunately, without a priority on training, the more skilled maintainers are drawn deeper into a mode of not giving training or not honing their own skills with training.

9.2 What Needs To Be Trained?

9.2.1 Training - The Weapon System/Mission And Major Components

It is important to keep in perspective the reason why your OFP support organization is in existence. The OFP is an integral part of a specific weapon system which has a specific mission. Also of significance are the major components of the weapon system. Oftentimes this perspective is clouded because the OFP maintainer's training program has not been established as an integral part of the OFP software life cycle for that particular weapon system.

Also important is that OFP maintainers have a working knowledge of their weapon systems. This knowledge should include the features of the weapon system, the mission of the weapon system, and the associated sub-systems or components of the weapon system.

The features of a weapon system include its physical make-up, its capabilities, its crew, and its history. The mission of a weapon system is how the system is being, and will continue to be, utilized. The major components of the weapon system could include its radar, its electronic warfare systems, its armament, and its communication and navigation systems.

Without a continuously updated knowledge of these features of his environment, an OFP maintainer is limited in the scope of his ability to support the weapon system's OFP.

9.2.2 **Training - The Target Processor(s)**

An OFP, as stated earlier, is the software portion of an Embedded Computer System (ECS). The ESC is oftentimes a specially designed microprocessor which has been selected for its ability to reliably operate under adverse conditions. Design considerations for these ECSs have not until recently included ease of programming or reprogramming. Many ECSs have their own peculiar assembler language as well as unique hardware features. The OFP maintainer must be thoroughly familiar with this peculiar language as well as the various hardware features in order to optimally utilize this device in its weapon system's application.

9.2.3 **Training - The Support Environment**

9.2.3.1 **Training - The Dedicated Computer Systems**

OFP maintainers utilize their support environment's dedicated computer systems to design, optimize, test, debug, document, train, and measure their OFP software. Dedicated computer systems are seldom easy to learn and are continuously being updated. The OFP maintainer is most efficient when he can utilize all the features of his dedicated computer system. This takes thorough up-front training and continuous up-date training as the dedicated computer system matures.

9.2.3.2 **Training - Hardware (Terminals, Printers, Disks, Etc)**

The OFP Maintenance Environment includes a variety of hardware items which the OFP maintainer must utilize to perform his duties. These include computer terminals, printers of various types, disk and tape drives, hardware interfaces, and simulators to name a few. Many hardware items have been specially developed for their environment, and require specialized training for proper utilization.

9.2.3.3 **Training - Software (Control Language And Utilities)**

The operating system has its own control language, which the OFP maintainer must understand in order to work on his projects. As the OFP Maintenance Environment absorbs new tools and methodologies, the maintainer has to stay on top of these software implementations as well. In many OFP environments, specialized utilities are developed in order to ease redundant or complex tasks. Oftentimes these are developed in-house and require the originator to develop training for their continued application.

9.2.4 **Training - Documentation**

Unfortunately, documentation is oftenleft as an afterthought. OFP maintainers would benefit greatly if their formal training included a method and a disciplined approach to documenting software.

The training of documentation should include an oversight into all the software life-cycle areas which require documentation. This would allow the maintainer to appreciate the value of documentation at his part of the software life-cycle, as well other areas he often does not interact with directly, such as requirement reviews, design reviews, and final system integration and test.

9.2.5 **Training - System Conventions**

System conventions must be understood, practiced, and updated periodically in order to insure the proper, secure, and safe operation of the OFP Maintenance Environment. OFP maintainers need to be held accountable to these conventions after thoroughly understanding the convention's implementations.

9.2.6 **Training - The Simulation Environment**

9.2.6.1 **Training - Overview Of Simulation**

A means needs to exist by which a OFP can be proven out. This allows for static and dynamic testing, and debugging. A simulation environment is such a means. Most simulation environments are very complex as well as very specialized. The designers and builders of the simulation environment should be required to include a method of training as well as good documentation as they implement simulations. The OFP maintainer not only should understand the simulator, but have the skills required to operate that simulation, such as flying, radar operation, weapon utilization, etc.

9.2.6.2 **Training - Simulations/Emulations/Workstations**

The OFP maintainer needs to understand the environment and the conditions that his software will operate under. Ideally, he would use the actual integrated system in its actual environment. Practically, he uses simulations to mimic his required environment and conditions. Simulations use the hardware (LRUs, displays, controls, etc) of the system as much as possible and use software to represent those areas which are unavailable.

As an extension of simulation, emulation allows complex systems to be copied in the form of software models. This is becoming more attractive as spare aircraft Line Replaceable Units

(LRUs) become more difficult to procure. Engineering workstations are providing the local computational power to make emulations more of a practical reality in OFP simulators. The OFP maintainer must be skilled in using the engineering workstations as well as understanding the emulations.

9.2.7 **Training - The Modules/Functions Of The OFP**

The OFP maintainer needs to have a knowledge of each of the modules and the functions of his OFP. A function of an OFP might be navigation, over-load-warning, or air-to-air radar mode. Modules are a section of an OFP which perform a part of (or all of) a function.

In a complex OFP, a maintainer might specialize in a few of the OFP's modules which allows him to more efficiently maintain those areas. For example, one maintainer might be responsible for all the modules which deal with missiles. He would have to understand the intricate details of each missile system, their complex algorithms, and the interactive details associated with the other modules of the OFP.

As OFPs become larger and more complex, the maintainers will have to cover a larger number of modules which are increasingly integrated. The maintainer will be forced to relinquish dependence on his own expertise, and use continuous training to support the dynamically changing OFP modules he is responsible for.

9.2.8 **Training - The OFP Change Process, A Run-Through**

One excellent method for training new maintainers the OFP change process is by creating exercises which require following all the steps necessary to carry out real OFP changes. This can be as simple as having a single variable changed and loading the change up to be observed. Complexity can be added to the exercise by introducing constraints such as timing, memory, or multiple variables to represent real OFP situations.

9.2.9 **Training - Diagnosis/Analysis/Isolation/Integration/Test**

Complex skills required to maintain OFPs are the diagnosis of problems and change requests, the analysis of the resources required to make a change, the isolation of faulty software logic, the integration of multiple software changes, and the design and implementation of detailed OFP testing.

Given a clearly stated requirement for OFP change, maintainers have to know how to implement that change and what resources are required to enable their implementation including memory, man hours, integration time, and testing time.

A simple OFP change might be a change closely related to a past change, and thus easily performed. A complex change might require that the maintainer obtain specific training, alter large amounts of code, design specialty test scenarios, and spend many hours integrating and debugging the change.

10.0 HOW DO WE MEASURE OFPs?

What metrics would help OFP maintainers and managers better understand the cost and complexity of their tasks? Usually the most quoted metric is "Lines of Code". Lines of Code does not account for more efficient coding or coding conventional type changes.

Attention needs to be paid to broadening traditional metrics such as Lines-of-Code by discussing other software support parameters such as OFP comparative analysis, software developmental research, maintainer skill level, software quality, and software reuse.

10.1 The Requirement For Metrics

Measuring the effort involved along with the complexity of OFP changes is a difficult but necessary task. A major effort in maintaining OFPs is the justification and procurement of the resources (man-power, dollars, and facilities) needed to carry out the processes of OFP updating, OFP testing, and OFP documentation.

10.2 The Baseline

The Baseline is an operational block cycle OFP selected for its nearest similarity to the new OFP coding effort. In many cases, OFP block cycle changes require minimal overall coding changes so configuration management between the Baseline and the new OFP can be maintained by actually using the Baseline to begin the coding effort. It is much easier to maintain closely related OFPs then a variety of peculiar ones. Baselines are also appropriate for documentation, tests, and training purposes.

10.3 Comparative Analysis

Comparative analysis is the process by which two or more similar software files are compared to discover the overlapping of the files. Non-overlapping code would be inserted code or deleted code. Further analysis might reveal added or deleted variables, documentation, or even unique modules. A manual comparative analysis is performed by examining the two or more files next to each line-by-line. Identical lines are marked off as such and differences are noted as well.

Comparative analysis utilities also exist in software form. These utilities vary in complexity and performance, but essentially automate the manual line-by-line analysis.

10.4 **Lines-of-code/Code-per-day**

The most commonly referred to software metric is the Lines-of-Code which can be further quantified as Code-per-Day. Lines-of-Code looks at the new source code generated between OFP versions. Code-per-Day takes Lines-of-Code and divides it by the period of time required to generate it.

These are crude metrics because they do not quantify the complexity of change or the externalities associated with change. Some OFP externalities would include skill level of maintainers, adequacy of support environment, patched code (pirated or reused bits and pieces of code), and project priority.

10.5 **Structured Programming**

Structured Programming aligns source code in easy-to-read and digest modules in a top-to-bottom configuration or a bottom-to-top configuration. The modules are designed to perform related tasks and use related variables. A well-structured module contains 50-100 or less lines of code. Modules are duplicated rather than called or sent to, as in the case of a FORTRAN GOTO statement. This could increase the coding effort and memory required, but drastically decreases the code complexity.

10.6 **Software Cost Analysis**

The OFP maintainer is increasingly called upon to identify the costs related to each phase of his OFP software development. The problem with this is that many software projects and resources overlap. A test plan or a complex algorithm might be used over and over again with slight modification. How do you attribute the original high overheads to later projects?

What value is placed on the skill level of the individual OFP maintainers? A senior engineer with an intimate knowledge of a complex system should be considered an invaluable asset.

In order to truly represent software costs, values have to be placed on the individual resources and processes used throughout a OFP's Life Cycle Development. These resources and processes must be carefully differentiated between OFP block cycles to properly allocate their individual project value.

Charles P. Satterthwaite/Engineer
Software Concepts Group
(513) 255-3947

APPENDIX A

GLOSSARY

Acceptance Test Procedure (ATP) A sequential test which checks for the proper functioning of an Operational Flight Program (OFP) integrated with its Embedded Computer System (ECS) and its associated hardware.

Air-to-Air A radar mode which accommodates aircraft to aircraft engagements. Eg: dogfight, refueling, and Identification Friend or Foe (IFF)

Air-to-Ground A radar mode which accommodates aircraft to terrestrial engagements. Eg: bombing and strafing

APG-63 96K Radar A Hughes Air Superiority Radar System used on the F-15 Fighter.

Assembler A software development tool which compiles programmer developed source code into machine executable object code.

Automated Documentation A process by which supporting documentation for software is generated automatically when software code is manipulated.

Automated Tests A process which remotely controls the sequential checking of an OFP integrated into its ECS.

AUTO-VAL An Ada Software Tool developed at Wright Laboratory which automatically tests OFPs using the shared memory and remote control.

Avionics Integrated Support Facility (AISF) A building or part of building along with its dedicated computer system(s), simulator(s), support software, and personnel which support as ECSs.

Avionics Suite A complete compliment of electronic hardware and software used on an aircraft.

Baseline The most recent or nearest configuration of an OFP (or its supporting tests or documentation) which can be used to code up a new configuration.

BASEREG A software tool developed at Warner Robins F-15 AISF which checks the proper utilization of the F-15 Central Computer's Base Registers.

Block Cycle A fielded and accepted configuration of an OFP.

Break X A flashing display which warns a pilot of potential airborne hazards.

Central Computer (CC) An ECS on an aircraft weapon system which processes the integrated avionics suite of that system's operational environment.

Centralized Software Support Activity (CSSA) See Avionics Integrated Support Facility.

Classified Tests That part of an OFP Test which is considered Confidential, Secret, or Top Secret.

Cockpit Controls and Switches The actual instruments a pilot uses to fly his aircraft and interact with his environment.

Code-per-Day The amount of OFP Source Code that is produced in one day of coding effort.

Coding Convention The conventions such as configurational management, security, and safety which are adhered to when developing OFP Source Code.

Comparative Analysis A line by line comparison of similar OFP sets or subsets.

Compiler See assembler.

Concurrent Development The process by which two or more OFP development efforts occur simultaneously.

Configuration Management A process by which OFP coding efforts follow predetermine formatting in order to promote structured programs, code reusability, and supportability.

Controls and Displays The pilot's instrumentation as well as the visual displays projected to him.

Cost Analysis The process of determining the value of OFP Coding Efforts.

Debug The process of finding and fixing faulty software logic.

Dedicated Computer System A computer resource outside of weapon system which supports an ECS on that weapon system.

Documentation The process of making a record of the actions taken and reasoning for those actions of an OFP coding change.

Drone An unmanned air vehicle.

Dynamic Display A visual representation of a time-dependent event.

Dynamic Environment A real or simulated environment which includes time.

Electronic Warfare (EW) The defensive capability of a weapon system.

Embedded Computer System (ECS) A computer system which is onboard a weapon system.

Emulation The art of representing (with hardware, software, or a combination of hardware and software) a part of a weapon system's environment.

Engineering Change Request (ECR) A formal design request to change some part of an OFP.

Engineering Workstation A powerful personal computer which allows engineers to develop, simulate, and test OFP.

Enhancement A change which adds or deletes features which makes an OFP tailored for its mission.

Executive A module of an OFP which directs that program's timing and priority scheduling.

F-15 The McDonald Douglas Air Superiority Fighter Weapon System.

Fight and Fly The capability of a pilot to carry out his mission.

Fire Control Computer See Central Computer.

Flight-line Data Entry The process by which an aircraft is loaded with information, such as the loading of its OFP into its Central Computer.

Flight Test Testing an OFP by actually loading it into its ECS and flying it through some test scenario.

Flow Diagram A graphical representation of software logic.

Freeze To hold a running OFP on one frame of time.

Full-up Simulation A simulation which completely represents what an OFP will encounter when it is operational.

Function One of the areas an OFP that performs processes such as navigation, control and display, or Air-to-Air.

Functionality A measure of the performance of some set or sub-set of an OFP.

Gun Mode A function which enables the gun of a weapon system.

Heads Up Display An instrument mounted at the pilot's eye level which displays all the critical flight data while enabling the pilot constant outside situation awareness.

Historical Documentation All the records including Technical Orders, Version Description Documents, engineer's notes, flow diagrams, etc which accompany an OFP.

Ideal Documentation Completely descriptive, concise, and perfectly traceable documentation which fully supports an OFP.

Initial Condition Some aerodynamic condition such as altitude, air-speed, or target range which is preset for simulator testing of an OFP.

Initialize To load a OFP into its ECS and clear previously calculated mission data.

Integrated Testing Testing which couples two or more processes together such as software with hardware; the CC OFP and the Radar OFP; or the addition of parameters to processes whose outcome is established.

Integration Adding two or more processes together.

Interactive Testing The capability of an OFP Maintainer to observe various levels of his software while his OFP runs in its real or simulated environment.

Interrupt The halting of some OFP process during run time to address a higher priority process or event.

Instruction Set The machine level language of an ECS which empowers the processor to perform its tasks.

Life Support Those processes which deal with maintaining the life of weapon system's aircrew.

Line Replaceable Unit (LRU) A piece of electronic equipment housed in a portable container which is part of an aircraft weapon system.

Lines-of-Code A crude measurement of software efficiency.

Linking The coupling of compiled sections of software.

Lock-on The process by which a radar finds and holds some airborne or ground target.

Logic Flow The path by which software follows.

Low Level Testing Testing which checks the validity of small sets of variables or some simple process.

Machine Level Code Binary (1's and 0's) code which enables an ECS.

Mainframe Computers A large capacity processor which traditionally has been required to perform complex real-time simulations.

Mentor An experienced OFP Maintainer who trains Junior Maintainers how to support OFPs.

Metrics Measuring tools which gauge the efficiency and cost of OFP Software.

Mission The range of situations a weapon system must be able to handle.

Modules A subset of an OFP which handles specific areas such as navigation or controls and displays.

Munitions Missiles, bullets, rockets, or other ordinance carried on a weapon system.

Navigation The art of keeping track of your present position.

Networking Coupling two or more processing systems together so that they can receive and send each other information.

Operational Fully tested and proved, ready to be used as designed.

Operation FLight Program (OFP) The software of an ECS.

OFP Change Process A means by which OFPs are corrected and updated.

Operational Test and Evaluation (OT&E) The formal proving out of OFPs in their weapon systems through flight testing on test ranges.

Optimize To maximize the performance of an OFP given its design constraints (memory limitations, timing issues, processing power, etc).

Overload Warning System A module of an OFP which monitors an aircraft's stresses and reports them to the pilot.

Partitioned Memory The sectioning of a computer's memory resources so that they can be read from or written to.

Patch Software A piece of software developed for testing an area of an OFP such as a radar patch which presets radar inputs.

Peculiar Support Utility Software tools developed in support facilities which are required to maintain the OFPs. Egs: sort routines, flow chart tools, patch software

Periphery Interface A device which actively communicates with an ECS.

PROBER A OFP utility developed at the Warner Robins AFB's F-15 AISF and used to explore the full utilization of logic paths on the F-15's Central Computer.

Processor Peculiar Tests Specialized testing for unique features of ECSs.

Programmable Monitor And Control (PMAC) The capability to observe and control the data-flow in OFP simulations.

Read Only Memory (ROM) Memory which has been set so that it can only be read, such as a processor's instruction set.

Real-time The actual interaction with the environment as it is happening.

Reel to Reel Drive A computer device for holding magnetic tape media.

Requirement The needs of the stated mission, which drive OFP changes.

Reticle A pilot's display which facilitates the aiming and firing of a weapon system's gun at a target.

Shared Memory A computer's memory resources which can be accessed by one or more other computers.

Signal-to-Noise Ratio (SNR) A measure of the effective strength of a signal with relationship to its surroundings.

Simulation Environment The use of hardware/software/both to provide the capability of testing ECSs while controlling the test parameters.

Software Engineer An individual who applies the disciplines of formal engineering to the development and management of software.

Software Life Cycle The period of time which encompasses software's original requirement, design reviews, coding, integration, and testing as well as reuse.

Software Quality A measure of the efficient utilization of software for its specific mission.

Specialized Test A OFP Test developed specifically to test an attribute of the software such as Built-In-Test (BIT) or Overload Warning System (OWS).

Software Reuse Sections of OFP software which can be utilized as is in other OFPs.

Static Display Displays which are not time dependent and when enabled are constantly on.

Structured Programming A method of programming which follows top to bottom (or bottom to top) logic flow, contains similar variable and processes, and constrained to 100 lines of code or less.

Sub-function A set from a function which performs closely related tasks.

Sub-system Failure The failure of one or more of a weapon system's avionic units such as a radar failure or an Inertial Navigation Unit failure.

Support Environment All the resources required to develop and maintain a system or a sub-system.

System Convention The rules and regulations which are followed in developing and maintaining OFPs.

TAC Form 37 A Tactical Air Command Software Engineering Change Request for OFPs.

Target Processor The ECS which houses an OFP.

Technical Description Document (TDD) A very detailed description of all the changes made in an updated OFP.

Technical Orders (TOs) A detailed manual describing the design, operation, and maintenance of a weapon system and its sub-systems.

Terrain Following Radar A radar system specifically designed to allow a pilot to fly his aircraft at very low altitudes.

Threat Any environmental condition which effects the completion of an weapon systems's mission and the survivability of that system.

Throughput The timed rate at which information is processed in an ECS.

Trade-off Analysis The science of selecting the best option given a set of limiting parameters.

Version An OFP which has been updated between Block Cycle Changes.

Version Description Document (VDD) A record of the changes made between versions or Block Cycles.

Weapon System A stand-alone system which can perform military missions. Eg: F-15 Fighter, M1A Tank, and Apache Helicopter

Wrung Out The thorough testing and correction of OFP logic bugs.