AFIT/GSS/LSS/93D-2

AN ANALYSIS OF THE ROOT CAUSES OF DELAYS AND DEFICIENCIES IN THE DEVELOPMENT OF EMBEDDED SOFTWARE FOR AIR FORCE WEAPON SYSTEMS

THESIS

Jay R. Hopkins, Captain, USAF
Curtis R. De Keyrel, First Lieutenant, USAF
AFIT/GSS/LSS/93D-2

The views expressed in this thesis are those of the authors and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

AFIT/GSS/LSS/93D-2

AN ANALYSIS OF THE ROOT CAUSES OF DELAYS AND DEFICIENCIES IN THE DEVELOPMENT OF EMBEDDED SOFTWARE FOR AIR FORCE WEAPON SYSTEMS

THESIS

Presented to the Faculty of the School of Logistics and Acquisition Management
of the Air Force Institute of Technology
Air Education and Training Command
In Partial Fulfillment of the
Requirements for the Degree of
Masters of Science in Software Systems Management

Jay R. Hopkins, B.S.
Captain, USAF

Curtis R. De Keyrel, B.E.E
First Lieutenant, USAF

December 1993

Acknowledgments

We would like to thank all of the people in the B-2, C-17, F-16, and F-22 SPOs that took the time to respond to our questionnaire. We are especially indebted to the members of these SPOs that allowed us to interview them for our case studies. Without their openness and honesty, we would not has been able to produce either an accurate or valuable product.

Our technical thesis advisors Captain John Robinson and Captain Dawn Guido also need to be commended for keeping us going in the right direction, and not allowing us to forget the main purpose of the thesis. We are also indebted to our third advisor Dr. John Muller for taking us on when no other permanent member of the Logistic School would.

We owe the most to our wives, Kelly and Shelly, who suffered along with us throughout this thesis process, but it serves them right since they both earned masters degrees without doing a thesis. Thanks for proof reading and listening to all our hours of bitching.

Curt & Jay

Table of Contents

List of Figures

List of Tables

Abstract

The importance of embedded software, used in every subsystem of all major weapon systems used by the United States Air Force, has increased drastically over the last decades. However, in spite of the regulations currently in existence, developing and acquiring software which meets the user requirements within the original cost and schedule estimates continues to be difficult. At the same time, the Air Force has pushed to improve the development process with the Total Quality Management (TQM) program. The primary method used to improve the process has been to create metrics, collect data on these metrics, and then perform a statistical analysis on this data. This process has resulted in large quantities data, but very little improvement. This thesis executes the forgotten first step of process improvement – to analyze the process and determine the significant problems faced while developing software for embedded systems. This goal was accomplished by examining and evaluating four major acquisition programs: the B-2, C-17, F-16 and the F-22. In each of these programs, the problems identified are categorized as either procurement practice, software development process, or personnel issues. Of these, procurement practices caused at least as many problems as deficiencies in the software development process.

AN ANALYSIS OF THE ROOT CAUSES OF DELAYS AND DEFICIENCIES IN THE DEVELOPMENT OF EMBEDDED SOFTWARE FOR AIR FORCE WEAPON SYSTEMS

I. Thesis Overview

1.1 Introduction

Embedded software is used in nearly every subsystem of every major weapon system in use in the United States Air Force today. The use of embedded software has increased drastically over the past twenty years. The F-111 accomplished 20% of its functions using software in its original design, in contrast to the B-2 which accomplishes 80% of its functions with software (Cannan, 1986:49).

The importance of embedded computers and embedded software has also increased over the years to the point where mission critical decisions have to be made instantaneously with little room for error. This situation is especially critical for aircraft which rely on embedded computers for the basic performance of not only their weapons and avionics, but for flight control as well. Since many aircraft in the Air Force inventory are designed to be aerodynamically unstable in order to maximize their performance, they can not be operated without the use of "fly-by-wire" flight control systems.

The criticality of these embedded computer systems has led to the establishment of rigid acquisition and development requirements. However, in spite of the regulations currently in existence for the Engineering and Manufacturing Development (EMD) phase,

there is an increased difficulty in developing and acquiring software which meets the user requirements within the original estimates of both cost and schedule. This problem is so serious that software often ends up as the pacing factor in the development of major systems. In other cases, software developed during the earlier Demonstration and Validation (Dem/Val) phase does not meet the same standards required as software developed in EMD phase, resulting in other problems.

1.2 Specific Problems

The system engineering process places little emphasis on software development until the EMD phase. For some major weapons systems such as aircraft, some software modules such as those required by the flight control system must be finished in order to complete the Demonstration and Validation phase. These prototyped software modules were produced without the restrictions placed on the software development process found in the EMD phase. This can lead to problems in the future since the contractor most likely eliminated many of these restrictions in order to minimize cost. It is also unlikely that the contractor will recomplete the work correctly, since the module has already proven to be successful. This lack of restrictions found in the Dem/Val phase can result in poorly documented code, causing it to be unmaintainable. Other problems, such as cost overruns and schedule delays, are similar to those encountered with software developed during the EMD phase.

Our hypothesis is that software developed before and during the EMD phase have some problems in common and other problems that are unique. Therefore, the research question we plan to study is "What factors result in inadequate software, cost overruns, schedule delays, and other problems during the development of software in both the Dem/Val and EMD phases?"

1.3 Objectives

The objective that needs to be accomplished is to determine what problems the software development process is experiencing on a regular basis. A tentative hypothesis for this objective is that the root causes of the problems attributed to deficiencies in the software development process are both internal and external to the software acquisition and development process. This hypothesis can be broken down into the following sub-hypothesis:

External root causes:

1. Low management priority toward software in the early stages of EMD.
2. Requirements creep.
3. Overall system development schedules are unrealistic.

Internal root causes:

1. Underestimates of time required for software testing and debugging.
2. Inadequate requirements analysis and review at major design reviews.
3. Lack of adequate metrics to measure software development progress.
4. Lack of adequate training for software management and engineering personnel.

1.4 Scope

One of the areas of embedded software that has experienced the largest growth over the last two decades is that of flight control software. Current fly-by-wire flight control systems have the same purpose regardless of the type of aircraft they are embedded into – to take an unstable aircraft and keep it in the stable flight regime with software control. Aircraft such as the F-16 and X-29 are unable to maintain controlled flight without such a flight control system, and every aircraft developed in the future will have a similar system to accomplish this purpose.

Aircraft such as the F-16 and the F-22 were prototyped to enter a fly-off competition and had the majority of their flight control software written during the Dem/Val phase. On the other hand, the C-17 and the B-2 followed the normal acquisition process and had

their flight control software developed during the original EMD phase. Therefore, data will be collected on these four aircraft that presently employ fly-by-wire flight control systems for this thesis.

The B-2 and the F-16 were chosen to serve as a baseline for a large and a small aircraft that were developed when digital fly-by-wire technology was new. The C-17 and F-22 are a large and a small aircraft that were recently developed using fly-by-wire technology. All of the above systems except the F-22 experienced delays and overruns in development attributed to software. Completing a case study of these four programs should pinpoint the problems that have existed over time and that effect both large and small aircraft. These programs are also representative of software developed in both the Dem/Val and EMD phases. By closely examining these problems, the root causes of the problems associated with the software acquisition and development process can be determined.

II. Literature Review

2.1 Introduction

This literature review will look at three areas which contain the background necessary to perform a comprehensive analysis of the problems in the Air Force software development process. First, the software development process will be summarized using the Mission Critical Computer Resources Management Guide and the relevant Air Force and DoD standards. Second, the specific area of flight control software will be reviewed from various papers by professionals in the flight control arena. Finally, historical problems in the software development of major weapon systems will be summarized from papers and articles written for professional conferences and congressional reports. The review of these three areas will significantly influence the development of the methodology used for data collection and analysis.

2.2 The Software Acquisition Process

2.2.1 The Overall Acquisition Process. Before looking at the specifics of the software acquisition process, it is useful to take a big picture view of the overall weapon systems acquisition process to see the environment in which the Air Force operates. The system acquisition life cycle consists of five phases as defined in DoD Directive 5000.1 and DoD Instruction 5000.2: concept exploration/definition (CE), concept demonstration & validation (Dem/Val), engineering & manufacturing development (EMD), production/deployment, and operations & support. Of these phases, the Dem/Val and EMD phases are of primary interest to this thesis, since these phases account for the majority of software development for new systems.

During the demonstration & validation phase, the various weapon system alternatives are definitized by evaluating the value and practicality of each alternative. This definitization is typically carried out using three methods: primary system hardware prototyping, paper studies, or a combination of paper definition and subsystem prototyping (McCarty, 1991:18). Since the goal of Dem/Val is to rapidly build a prototype at a low cost in order to establish the viability of the general system design alternatives, the majority of the more stringent acquisition and development regulations are typically not enforced. Since flight control software is often developed in this phase and not changed later, the Air Force is often stuck with unmaintainable software. This problem can greatly increase the cost and time required to make modifications or corrections to the software. It also greatly increases the risks of new problems being generated during the maintenance of the software.

The approval of Dem/Val at milestone II signifies the commitment to build, deploy and support the system by transitioning to the engineering & manufacturing development phase. EMD is the phase where most of the system is designed, developed, fabricated and tested (McCarty, 1991:20). This phase is also where the majority of the mission critical software is written.

Major modifications to system hardware can also signify a change in the mission critical software. Such a modification is initiated by a milestone IV decision which is similar to starting a new EMD phase.

2.2.2 Standards and Regulations. There are several military standards and regulations which govern the acquisition and development of embedded computer resources. However, these documents are continually being consolidated and revised. The basic premise of the governing documents has remained the same, but the location of the information has changed. The historically significant documents were described in detail in both the MCCR Management Guide and a thesis written by Robert Buckley at the

Naval Postgraduate School. The following is a summary of the documents that initiated some of the most important ideas:

DoD Directive 5000.29 - *Management of Computer Resources in Major Defense Systems, 26 Apr 76.* The purpose of this document was to establish a DoD policy for the management and control of computer resources during the life cycle of major weapon systems. This directive was the first major step taken by the DoD to address the growing software development problem, and it represented the first formal recognition of the criticality of software. The major purpose of this directive was to force software to be managed as a configuration item, similar to the management of hardware.

This basic directive caused other documents to be generated within each of the services to implement the directives enclosed. For the Air Force, these guidelines include:

AFR 800-14	*Life Cycle Management of Computer Resources in Systems, 29 Sep 86*
AFSCP 800-14	*AFSC Software Quality Indicators: Management Quality Insight, 20 Jan 87*
AFSCP 800-43	*AFSC Software Quality Indicators: Management Insight, 31 Jan 86*
AFSC/AFLCP 800-45	*Software Risk Abatement, 1988*
AFSCP 800-5	*Software Independent Verification and Validation, 20 May 88*
ASDP 800-5	*Software Development Capability/Capacity Review, 11 Sep 92*

DoD Directive 3405.2 - *Use of Ada in Weapon Systems, 30 Mar 87.* This directive established DoD policy for the use of Ada as the single common higher order language in the development of "computers integral to weapon systems" (DSMC, 1988:4-4). DoD Directive 3405.1 then modified 3405.2 to add languages for other DoD uses.

DoD Standard 2167A - *Defense System Software Development, 29 Feb 88.* This document is the keystone regulation for the entire software development and acquisition process. It sets the requirements to be used during acquisition, development, and support

of mission critical software systems. DoD Std 2167A defines the software development process to consist of eight major activities:

Systems Requirements Analysis/Design
Software Requirements Analysis
Preliminary Design
Detailed Design
Coding and CSU Testing
CSC Integration and Testing
CSCI Testing
System Integration and Testing

The following chart gives a general overview of the software acquisition process and major milestones and reviews as defined in DoD Std 2167A:

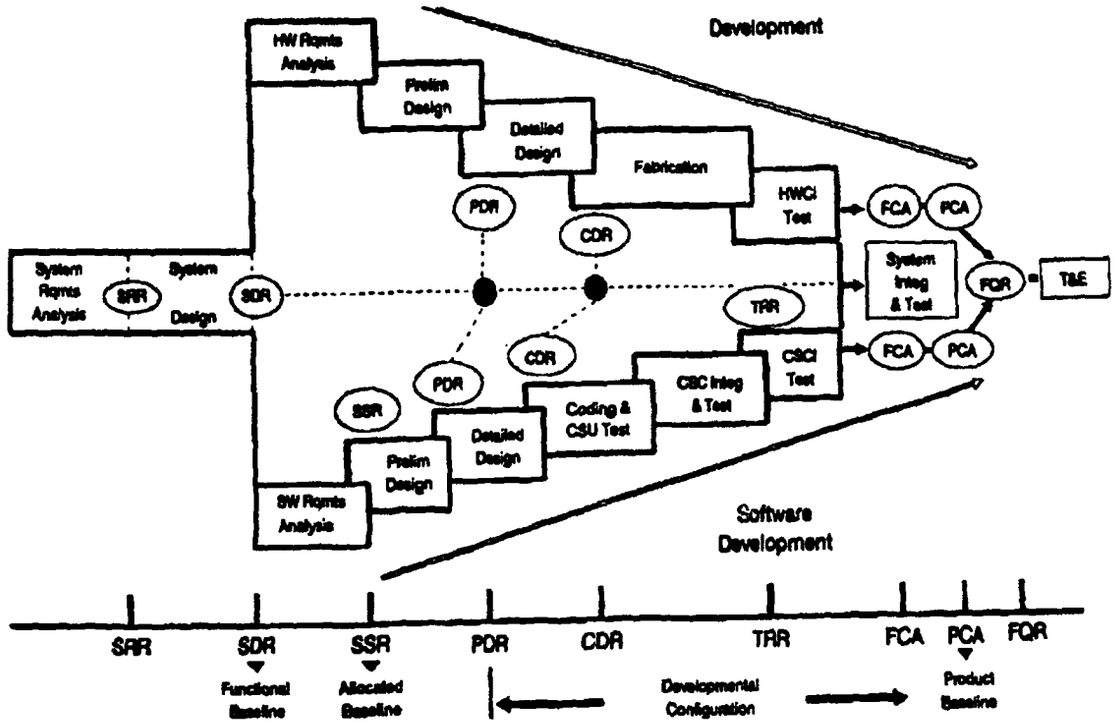


Figure 1. DoD Std 2167A Software Development Process

DoD Standard 2168 - *Defense System Software Quality Program, 29 Apr 88.* This standard complements DoD Std 2167A by establishing the requirements for the

development, documentation, and implementation of a software quality program. It also provides direction for the follow-up activities necessary for resolution of problems.

Mil-Std 1521B - *Technical Reviews and Audits for Systems, Equipments, and Computer Software, 1 Jun 76.* Establishes the requirements to be followed for conducting technical reviews and audits of computer systems and software.

Mil-Std 1803 (USAF) - *Software Development Integrity Program, 15 Dec 88.* Provides general requirements to achieve software integrity during development and deployment. It is intended to improve the performance and supportability of Air Force software.

DoD Instruction 5000.2 - *Defense Acquisition Management Policies and Procedures, 23 Feb 91.* Part 6 Section D - "Computer Resources," not only replaces DoD Directives 5000.29 and 3405.2, but also adds new policies for the procurement of software. The new policies deal with the use of software engineering practices during the development and acquisition of software. It defines the minimum acceptable set of practices to be used, and emphasizes how important it is to have a good software engineering development process.

2.2.3 Conclusions. Although many other documents affect the acquisition and development of mission critical software, those mentioned above have had the most significant impact to software specifically. In addition, many of the other relevant documents are referenced by the documents referred to above. Therefore, this summary provides a fairly comprehensive compilation of where to look for guidance in the development and acquisition of DoD software.

2.3 Flight Control Systems

Fly-by-wire flight control systems are being used in most aircraft that have been produced during the last decade, and all aircraft produced in the future will have either fly-by-wire or fly-by-light flight control systems. Fly-by-light flight control systems are the same as fly-by-wire except fiber optic cables are used to send the signals between the flight control computer and the flight control surfaces. This thesis will use the software development process associated with producing these flight control systems as a starting point in comparing the software development of four different aircraft. Therefore, a minimal understanding behind the principles of flight control systems is necessary in order to determine how this process relates to the development process for other systems. Three different items must be understood in order to have a good introduction to fly-by-wire flight control systems (FBWFCS):

1. The aerodynamic principles that require fly-by-wire flight control systems.
2. The performance benefits achieved by using FBWFCS.
3. How the Flight Control Computer controls the aircraft.

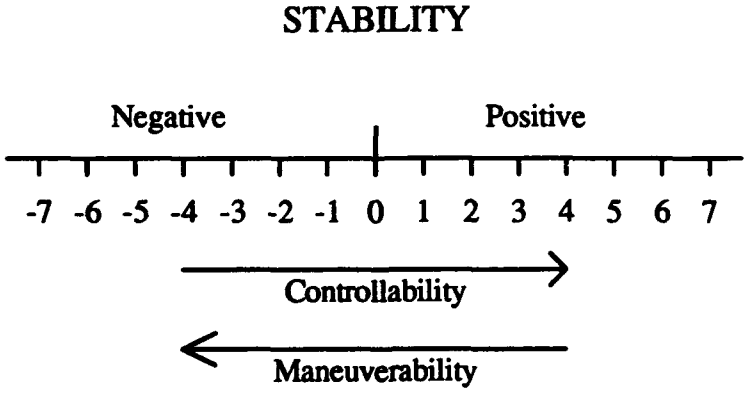


Figure 2. Static Stability's Effect on Aircraft Control

2.3.1 Aerodynamic Principles. In order to maintain controlled flight, all aircraft must be both statically and dynamically stable. Stability can be thought of as a number line similar to the one in figure 2 that has zero as its center. To the right of zero an aircraft has positive stability, to the left negative stability. Positive stability provides the aircraft with the ability to maintain a uniform flight condition and the capability to recover from unusual flight conditions. As stability is increased, the controllability of the aircraft also increases. However, maneuverability decreases as stability is increased. When designing a new aircraft, the designers try to minimize the stability of the aircraft, in order to maximize maneuverability, to the point that controllability is at the minimum acceptable standard required for the plane. The purpose of the flight control computer in a FBWFCS is to serve as an interpreter between the pilot and the flight control surfaces. When the aircraft is in an undesirable stability state, the flight control computer will change the signals it sends to the flight control surfaces in order to have the aircraft emulate a greater stability.

2.3.2 Definitions. Static stability of a system is defined by the initial tendency of an object to return to equilibrium conditions following a disturbance from equilibrium. If an object is disturbed from equilibrium, and has the tendency to return to equilibrium, positive static stability exists. If the object has a tendency to continue in the direction of the disturbance, negative static stability exists. If the object subject to a disturbance has neither the tendency to return nor the tendency to continue in the displacement direction, neutral static stability exists. These three categories of static stability are illustrated in figure 3.

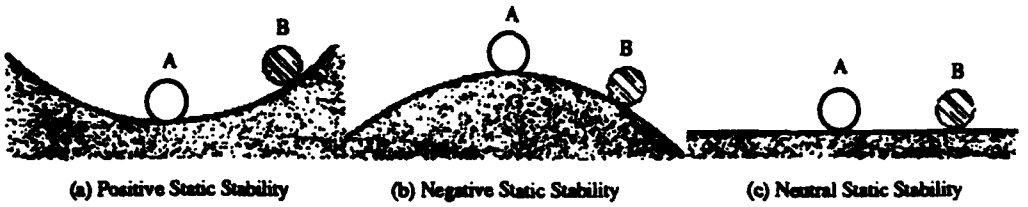


Figure 3. Static Stability (88th FTS, 1988:7-3)

2.3.3 Dynamic Stability. While static stability is concerned with the initial tendency of a displaced body to return to equilibrium, dynamic stability is defined by the resulting motion with respect to time. If an object is disturbed from equilibrium, the time history of the resulting motion indicates the dynamic stability of the system. In general, the system will demonstrate positive dynamic stability if the amplitude of motion decreases with time.

Dynamic stability is associated with two types of motion, non oscillatory and oscillatory. Figure 4 shows the three types of non-oscillatory motion and the types of stability associated with each.

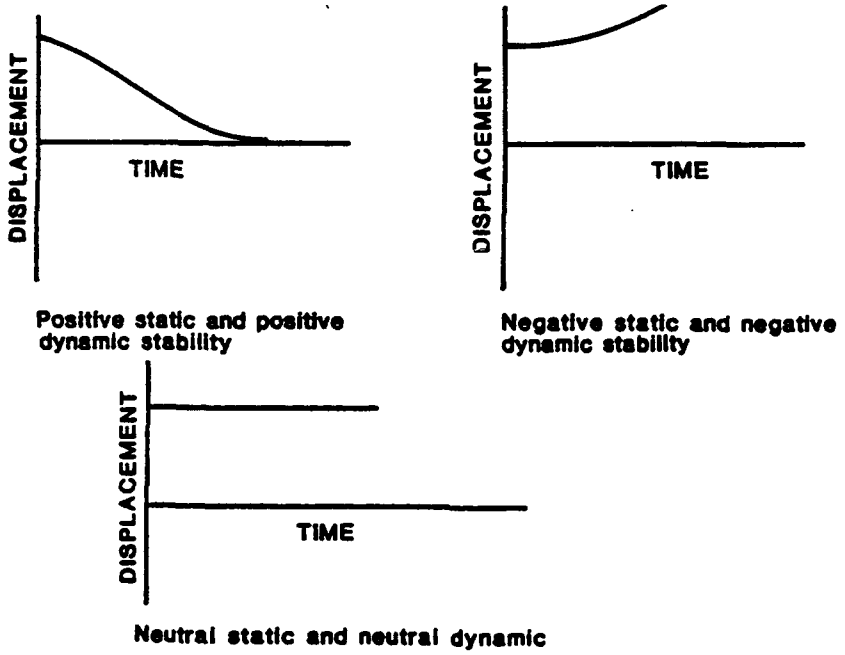


Figure 4. Non-Oscillatory Motion (88th FTS, 1988:7-4)

When the motion is oscillatory, positive static stability exists because the initial tendency after displacement is toward equilibrium. Figure 5 shows some typical examples of oscillatory motion.

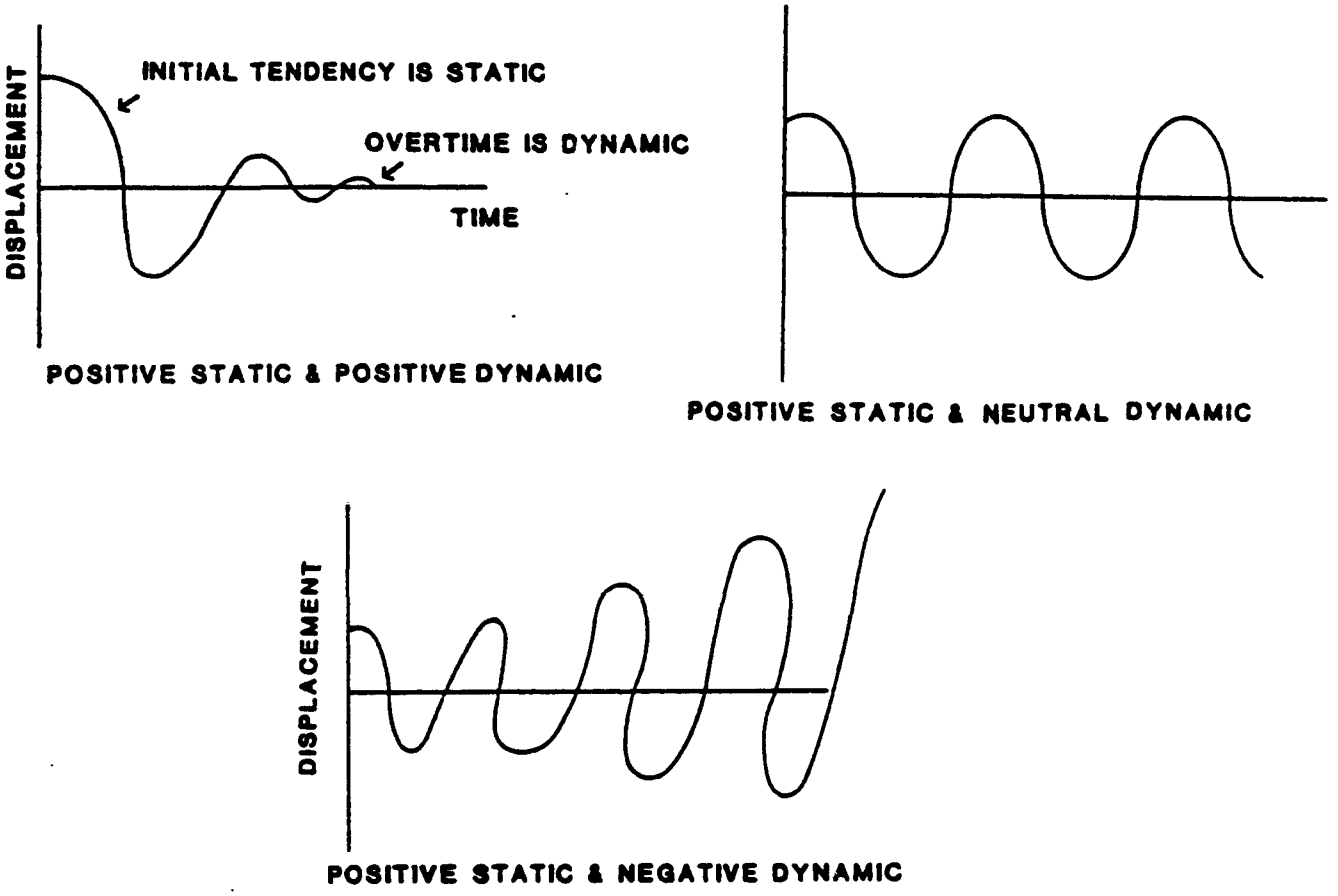


Figure 5. Oscillatory Motion (88th FTS, 1988:7-5)

All aircraft must demonstrate a required degree of both static and dynamic stability. If the aircraft is to have satisfactory flying characteristics, it must be statically and dynamically stable, or have a computer system to enhance the stability. When an aircraft is in an unstable flight regime, it will depart controlled flight and the pilot will not be able to provide proper inputs to facilitate a recovery to controlled flight. However, it is

possible with flight control computers to have the aircraft performing in the unstable flight regime, but remain in controlled flight.

All aircraft movements can be described as a combination of rotations around the three axes of rotation found in figure 6. Each of the three axes of an aircraft has a stability coefficient associated with it. The lateral and the vertical axes are naturally stable. However, the stability coefficient associated with the longitudinal axis, or pitch axis, is usually made as small as possible in order to maximize maneuverability. The following discussion will concentrate on the longitudinal axis and pitch stability, since they are the most relevant.

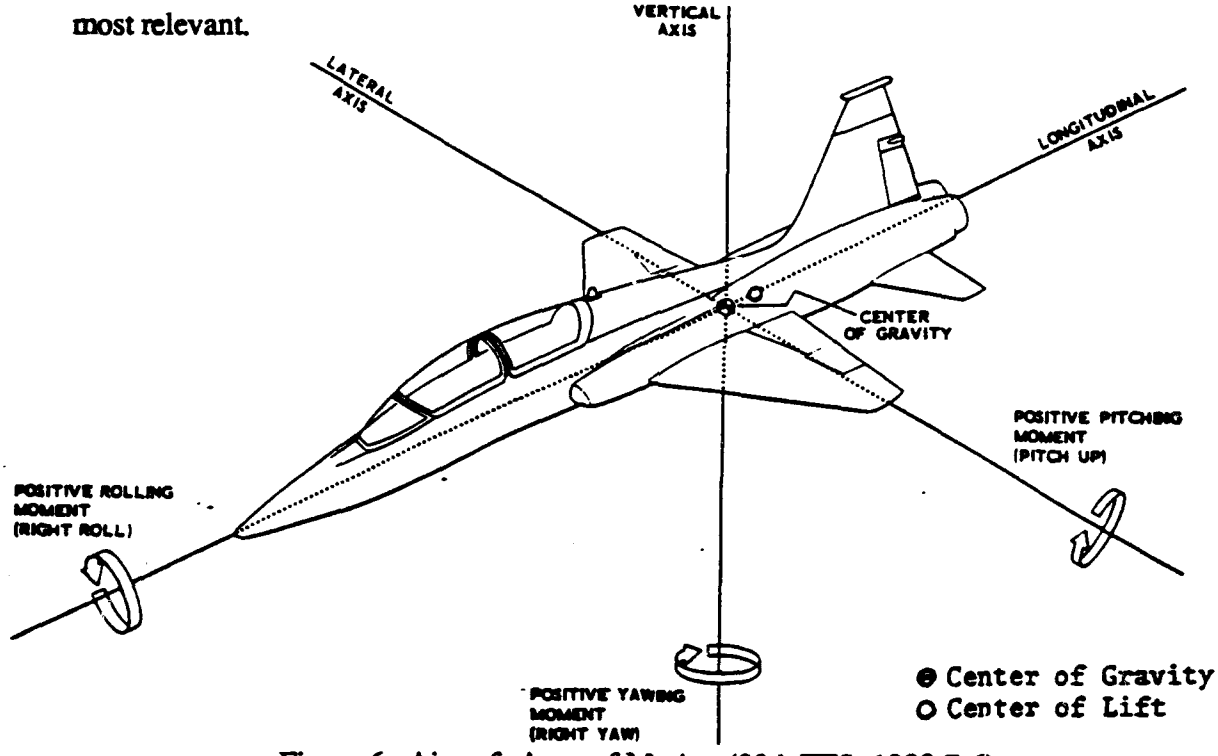


Figure 6. Aircraft Axes of Motion (88th FTS, 1988:7-6)

To analyze the design characteristics which contribute to pitch stability, the center of lift relationship to the center of gravity (CG) must be examined. If the center of lift is aft of the CG the aircraft is in the positive static stability region. Pitch stability is increased as the CG is moved farther in front of the center of lift. If this distance is great, the aircraft is very stable and wants to remain in its current state. Such an aircraft will be sluggish and

require large stick forces to elicit movement. As the positive stability is decreased the aircraft becomes more sensitive.

2.3.4 Performance Benefits. Changing the relative position of the center of gravity with respect to the position of the center of lift can have two beneficial effects. The first applies mostly to fighter type aircraft – moving the center of lift forward causes the airplane to become more maneuverable. Maneuverability has been the main design issue in developing new fighter aircraft since World War I. The pilot that has the most maneuverable plane should always win the battle when flying against a pilot of the same skill level in a plane that is less maneuverable. The basic concept of a dog fight between two fighters is to get the nose of your aircraft pointed at the opposing aircraft. At this point you can fire a weapon that will impact the other aircraft. This strategy has become less important as weapons have increased in technology. Today, an air to air missile can be fired before the other aircraft can be seen with the naked eye. However, maneuverability has remained the important criterion in designing new aircraft. Currently, maneuverability is also critical for defensive maneuvers to avoid these new high-tech weapons.

The second benefit applies mostly to large aircraft that carry a large payload, such as bombers and cargo planes. A large part of the cargo and payload bays in these aircraft is located behind the center of lift of the aircraft. By allowing the CG to fall closer to the center of lift, more cargo or a larger payload can be loaded into the back of the plane without decreasing the aircraft's ability to maneuver. This feature can improve the payload of the aircraft from an unacceptable level to a very beneficial one. The B-1B has experienced this restriction with its mechanical flight controls as stated in a Congressional Budget Office report.

> The B-1B is designed to fly at low elevations of 200 to 400 feet during a penetrating mission in order to avoid Soviet air defenses. During such terrain-following flights, the

> B-1B must have the ability to maneuver, including the ability to pull up sharply to avoid hitting hills. To maintain the ability to pull up at the level desired by the Air Force (2.4 g's, or gravitational equivalents, for 10 seconds), however, the B-1B can only carry about 125,000 pounds of munitions and fuel, which is significantly less than originally planned. This situation has occurred because the B-1B cannot, with its basic flight control system, fly at as high an angle of attack (the angle between the wing and the relative air flow) as anticipated, reducing the amount of weight it can carry. (Merkley, 1988:5)

This problem forced one of two solutions on the designers of the aircraft – get the Air Force to lower the payload requirement, or add a flight control computer to control the aircraft's flight control surfaces in order to allow the aircraft to fly in less stable configurations without sacrificing maneuverability.

The C-17 faced a similar situation, as an aircraft slows the angle of attack must be increased in order to provide lift equal to the total weight of the aircraft. Short field landings could not be performed at the required weight limit without reducing the approach speed to a point requiring the angle of attack that would put the aircraft into the unstable flight regime.

2.3.5 Software Implementation. The easiest way to envision fly-by-wire flight control system is as an interpreter between the pilot and the flight control surfaces. When the aircraft is operating in the stable flight regime, the flight control computer does nothing to the inputs it receives from the pilot and passes them to the flight control surfaces. However, when the aircraft is in the unstable flight regime, the pilot's inputs are translated into commands that will move the flight control surfaces in whatever manner is necessary to achieve the result the pilot wants. In some cases this can be the exact opposite of the movements that would have been required if the aircraft had been in the stable flight regime.

The flight control computer determines what actions it should perform on the pilot's input with its software procedures. The first procedure is constantly running to read sensors and determine the flight regime in which the aircraft is operating. The results of

this procedure are passed to another subsystem that determines how to translate the pilot's inputs into the correct movements of the flight controls for the current state of the aircraft.

2.3.6 Conclusions. This review has been a very simplistic look at flight control systems. However, it should be detailed enough to evaluate the data collected. It is important to remember that the problem of the software development process is the focus of this thesis. The development of flight control software should be representative of most embedded computer software projects. However, flight control software is considered a flight critical system, or a system whose loss results in catastrophic consequences for the aircraft and possibly the pilot (de Fio, 1988:15-1). Therefore, it has many redundant hardware and software features built into it in order to eliminate failures. This redundancy results in bugs being found early or never found at all. Most software projects do not have this luxury. This fact must be kept in mind in the analysis of the data.

2.4 Problems In Software Development

Most of the articles reviewed cite problems which could fit into three main categories: personnel, procurement practices, and the software development process. The problems which fall into these categories are of primary interest to answer the research question. The following is a brief discussion of the issues which fall into each of the three categories.

2.4.1 Personnel. The single most important factor in a successful software development/acquisition program is to have competent personnel. The first personnel problem starts at the top of the management chain – most program managers do not have any software experience. Some have experience in hardware, but many "government acquisition teams are being led by managers specializing in administration or finance, not the technical areas involved in the program" (SIO, 1989:20). The article goes on to claim that none of the managers or their staffs have "up-to-date" technical expertise in fields in

which they are working. How can we expect people to have an up-to-date understanding of software development when they do not even have the current information on the hardware they are dealing with in their project? The bottom line is that better educated upper management will be able to help identify difficulties before they become crises, and can also assure adequate development time, staffing, and resources (Hall, 1991:609).

Furthermore, software managers are often given their positions with no prerequisite experience or training in software disciplines. Software managers often lack experience and have never completed a structured software engineering education program. This lack of qualified personnel in key roles is due, in part, to the lack of available qualified personnel in government service (Hall, 1991:609).

In addition to the lack of software experience in upper management, software management, and engineering, there is also a lack of understanding of software issues by support personnel. Since software is required by the DoD directives to be acquired as a configuration item just like hardware, functional positions such as contracts, logistics and configuration management also need to know the nuances of software acquisition versus hardware acquisition.

Part of the reason for this lack of experienced software professionals in the government is the absence of adequate compensation, both financially and in job satisfaction, in software management, engineering and quality assurance positions. One example cited is that "today a GS-13 Personnel Specialist earns the same salary as a GS-13 Computer Scientist. This is not reflective of industry compensation rates" (Hall, 1991:609). Therefore, anyone with sufficient software training and experience in the government is motivated to move on to the greener pastures in the private sector.

Finally, personnel from the contractor side often suffer from a lack of understanding of software issues in upper management and lack competent personnel in software engineering and quality assurance. Software expertise is normally not a significant

consideration during source selection and, even if it was, there exists no formal standardized criteria for evaluation.

2.4.2 Procurement Practices. Problems exist in the overall procurement process which inhibit the success of the software portion of a program. First of all, the procurement process does not incentivize the use of good software engineering practices (Hall, 1991:608). Short term factors such as keeping cost low and meeting schedule often take precedence over more important long term factors such as reliability, adaptability, reusability, and maintainability. The government unintentionally encourages such behavior from contractors since cost and schedule are immediately measurable, whereas the other factors are not evident until well into the system's life cycle. As with hardware, unrealistic user requirements may contribute to a *rushed* schedule resulting in poor quality software, which has a greater negative impact later in the program (Hall, 1991:612).

Concurrent hardware/software development also causes problems in the software development. The specification and implementation of the hardware assemblies and the software that will control them are generally done concurrently in most embedded computer systems (Larman, 1989:706). Allocation of system requirements is made initially between hardware and software. Subsequently, the life-cycle paths diverge resulting in schedules which are rarely concurrent. For example, if sufficient progress is not made in the hardware area, there is no absolutely valid way of testing the software.

Likewise, changes in the user requirements may affect the concurrency of the hardware/software schedules. Often, when the hardware design is advanced, additional requirements result in changes to software rather than hardware since it is perceived that software changes will be cheaper than changes to the hardware design. It is also easier to justify a software change since the cost of a hardware change is easily quantified, whereas the cost of software changes may not be evident until the long-term software schedules start to lag behind the hardware. Usually, these types of requirements changes come later

in the program as the user becomes more involved. This user involvement leads to the next problem.

"Little user involvement in the early stages of program and requirements definition affects supportability as well as operational suitability" (Simmons, 1990:65). The later in the program a software change takes place, the higher the cost. This is the case whether the change is due to either a defect or a new requirement imposed by the user. The following table demonstrates the importance of recognizing these required changes early.

Discovery Phase	Increased Cost (times original cost)
Coding/Build	1.5 to 3
Integration	2 to 5
Validation	3 to 10
Flight Test/Operations	10 to 90

Table 1. Cost of Fixing Software Errors (Simmons, 1990:66)

Until software is considered at the beginning of the system development process, when hardware is first considered, project and procurement managers will discover as their programs slip behind schedule and go over budget that "software is now the choke point in large systems" (Reed, 1988:37).

As mentioned previously, the management focus on hardware has also been a significant drawback in the development of software. Hardware and software have both been progressing at exponential, though not equal, rates since computers were first developed. Hardware was very expensive and inefficient at first, but has expanded toward being inexpensive and very efficient. Software on the other hand has gone from an inexpensive and simple part of the system to a very complex and even more expensive part. "While software now drives system requirements, the procurement system still focuses attention on hardware" (SIO 1989:4). This situation is not surprising. The editors of Aviation Week & Space Technology have stated that "The procurement establishment is undoubtedly more comfortable dealing with hardware than software" ("Achilles Heel,"

1988:15). It is understandable that people are still more comfortable with hardware since it has been the primary focus of computer systems since they were first envisioned. Therefore, it has received the primary amount of attention. We must also remember that the computer resulted from years of research in the field of electrical engineering. Computer science on the other hand did not come into existence until the first computer was developed. Mills puts this idea into a very good perspective in his article "Engineering Discipline For Software Procurement."

> Although software is everywhere in electronics and communication systems today, we need to remind ourselves that software is, indeed, a new human activity, only a generation old. When civil engineering was that old, the right triangle was yet to be invented. (Mills, 1987:1)

2.4.3 Software Development Process. The software development process is relatively new compared to the hardware development process and is not as mature. Procedures and standards are well established for hardware development, but are highly dynamic in the software area. The Software Engineering Institute (SEI) as well as AFMC are developing methods to evaluate contractors processes for software development. Watts Humphrey, the main architect of the SEI's Capability Maturity Model (CMM), based the development of the CMM on Deming's principle that the product produced will only be as good as the process used to produce it (Humphrey, 1989:3). It is estimated that over 80% of all software organizations have an immature software development process. This lack of process has resulted in fragmentation and non-standardized development environments throughout the defense industry (Hall, 1991:614). This immaturity and fragmentation accounts for a portion of many of the other problems encountered in the software development process.

The immaturity of the software development process for many organizations has resulted in the systems engineering and software engineering functions remaining relatively

disjoint. Software and systems engineering processes must be integrated to develop a disciplined, manageable and optimizable system (Hall, 1991:615).

Measurement of contractor progress during the software development process continues to be a problem for the government. One of the major deficiencies in this area is the development of standard metrics which are collected for each program (Hall, 1991:615). Without these metrics, it is difficult to quantify software productivity and get an unbiased report of progress from the contractor.

In the current government procurement standards, "The acquiring agency also has few review points or measurable milestones during the development process" (Simmons, 1990:665). In fact, they are not well proportioned throughout the development. "The major review points in the development phase occur before development is 50% complete, with almost no review milestones occurring until program completion" (Simmons, 1990:665).

Even though most of the reviews are early in the program, "generally the most significant or difficult problems occur early in the development cycle but are not found until integration and test have begun" (Simmons, 1990:665). Many times these problems can be traced back to unfeasible or even nonexistent performance requirements, indicating the improper allocation system requirements as well as inadequate requirements analysis.

2.4.4 Conclusions. It is easy to see that these problems are not mutually exclusive of each other, and the three categories mentioned – personnel, procurement practices and the software development process – have many overlaps as well. However, it is useful to divide the problems into these categories for analysis purposes. These problems cited are by no means a complete listing of all problems in software development. This thesis will attempt to add to, or perhaps consolidate, this list through further study in this area.

III. Methodology

3.1 Introduction

During the last few years there has been a big push in the Air Force to improve the acquisition process through the Total Quality Management (TQM) program. The main method to accomplish this has been to create metrics, collect data on these metrics, and then perform a statistical analysis on this data. The logic to this method is flawed – any procedure that attempts to predict the capability of a process without first checking that process for stability will invariably yield a faulty picture of the process (Wheeler, 1986:135-136). Another problem with this approach is that people are now measuring everything without considering what information they need to answer their questions. Therefore, the methodology of this thesis is designed to not only determine the root causes of the problems in the field of software development, but to analyze these problems in order to determine which are significant. Of these significant problems, this thesis will try to determine which are stable so that they can be measured in the future and provide quality statistical information to the TQM process instead of meaningless data.

This goal will be accomplished by examining several different programs using a case study approach. The case study approach will include personal interviews with individuals working in each of these programs followed by a survey of the software personnel in each organization.

3.2 Population and Sample

The population for this thesis is all software development projects for aircraft embedded systems. It would be impossible to develop a sample that exhibited all of the

properties found in this large population. It would be equally difficult to develop a sample that is made up of mutually exclusive subsets of the population. Therefore, the sample chosen to narrow the focus of this thesis is aircraft that have similar flight control systems and their associated avionics groups.

Four subsets of the population were chosen for the study which have the relationship pictured below in figure 7. These four programs include the development of the flight control software for the B-2, C-17, F-16, and the F-22. The programs are interrelated in the following ways.

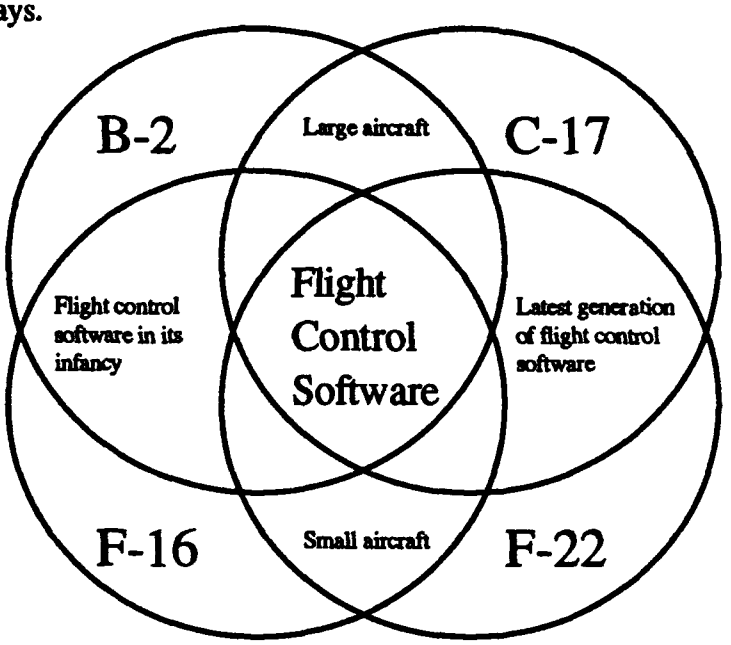


Figure 7. Relationship of Aircraft in Sample Population

All four aircraft have flight control software development projects in common. The B-2 and the F-16 had their flight control software developed when fly-by-wire technology was in its infancy. The C-17 and the F-22 are the latest generation of aircraft to have flight control computers installed. The C-17 and the B-2 are both large aircraft and have similar needs for flight control software. The F-16 and the F-22 both are small fighter type aircraft and have relatively the same mission and requirements for their flight control

software. By analyzing these sub-groups and their correlation, an accurate prediction about the population as a whole can be made.

3.3 Research Methodologies

Due to the relatively small number of aircraft which have been produced at the Aeronautical Systems Center (ASC) in recent years and the small number of software professionals involved in the development of their flight control software, a case study methodology is appropriate for this thesis. This case methodology will be supported by a survey of the experienced software professionals in each of the programs.

3.3.1 The Case Study Methodology. A case study is an empirical inquiry that:

- investigates a contemporary phenomenon within its real-life context; when
- the boundaries between phenomenon and context are not clearly evident; and
- multiple sources of evidence are used (Yin, 1991:23).

Case studies are generalizable to theoretical propositions rather than to populations or universes (Yin, 1991:21). Put more simply, it is the case study investigator's goal to expand and generalize theories, not to enumerate frequencies as in a traditional statistical study. This goal coincides with the goal of this thesis – discovering the significant problems in software development.

The specific methodology used in this thesis is a multiple-case design. Multiple-case studies are considered to be more robust due to the "replication" logic used. This approach is analogous to repeatedly conducting the same experiment to validate the results. Every case in a multiple-case design should serve a specific purpose within the overall scope of inquiry (Yin, 1991:48). Several sources of evidence will be used in each of the cases studied: interviews, documentation, and archival records.

3.3.1.1 Interviews. Individuals from each of the program offices will be interviewed on the details of their programs. This will be a structured interview where

open discussion will reveal the specifics of the software development history for each aircraft. Contractors involved with the aircraft's software development will also be interviewed if an agreement can be reached with the program office and the company.

General topics to be discussed in the interviews include program background, the software development process, and software development progress to date.

3.3.1.2 Documentation. Documentation exists in numerous repositories for each of the aircraft being studied. A few experts in the software area have worked on multiple aircraft, providing a wealth of experience as well as comparisons among different programs. Since engineering is matrixed to the program offices in ASC, this single organization is a significant source of documentation.

3.3.1.3 Archival Records. General Accounting Office reports exist for many of the aircraft developments programs, some of which specifically mention software. In addition, magazine articles are routinely written describing the status of the programs as well as any technological advances. Since digital flight control software is a relatively recent advance in technology, sufficient archival information should exist.

3.3.2 Survey Methodology. One method of enhancing a case study is the use of a survey. The use of a survey ensures both a common frame of reference for each of the cases and also ensures that certain basic information is gathered as objectively as possible. The individuals picked to complete the surveys will assure validity of the data. Each person will be identified through the contacts in the program offices to ensure that they are intimately familiar with the software on the particular program.

The survey should take less than ten minutes to complete. For the potential respondents that are located on Wright-Patterson AFB, we plan to take the questionnaire to them in person and ask them to fill it out while we wait. All other potential respondents identified will be mailed the survey with a self addressed stamped envelope. By using these two techniques we expect to maximize the number of respondents.

3.3.2.1 Initial Development. After a substantial list of problems is compiled through a literature review, an initial survey will be created to determine a rough cut at the most substantial ones. This survey will contain a list of problems that will be ranked according to the importance in software development. This survey will also contain blank sets of scales so the respondents can add and rank their own problems not found on the list provided. After this survey is created, it will be checked by a research methods expert for validity and bias.

3.3.2.2 Initial Testing. This initial survey will then be used on a select group of people that are knowledgeable in both software development and general program management. The results from this survey will be analyzed to determine a list of problems that these experts feel are the most important during software development and that have some degree of stability to them.

3.3.2.3 Final Development. The list of important potential problems will then be presented on a second survey. On this survey individuals will be required to rank the list of problems presented to them from most important to least important. It is important to note that the survey data will be part of the findings for each of the individual cases.

3.3.2.4 Final Testing. This final survey will also be presented to the research methods expert to be checked for validity and bias. It will then be tested on people that are knowledgeable in software development, program management, and survey methods. After their comments are reviewed and the data received from their survey has been analyzed, the survey will be revised to a final form. This final form will be presented to the research methods expert, and barring no problems will be the survey presented to the sample of the study population.

3.3.2.5 Statistical Tests. The test statistic used to measure the agreement of all the experts selected is Kendall's coefficient of concordance. This statistic is used when "we may be interested in the degree of agreement among several, say m (where $m > 2$),

sets of rankings of n objects or individuals" (Daniel, 1978:326). In the case of this thesis n potential causes of software development problems will be ranked by m experts on the selected programs. The rank of each root cause and the magnitude of the contribution to the overall problems experienced in the program will determine the most significant root causes.

In addition to the rankings and the relative contributions of each of the factors to the software-related problems of the program, it is also necessary to measure agreement amongst the individual experts' rankings. This measure is quantified by the test statistic found in equation (1) of section 3.4.4 Kendall's Coefficient of Concordance.

3.4 Analysis Tools

The objective of this thesis is to perform the first step in the statistical quality control process for the software development process. Therefore, a review of the important tools that deal with the statistical quality control process is in order.

3.4.1 Importance of Data. All texts on statistical process control agree that the purpose for collecting data is to have a basis to take appropriate action. According to Wheeler, the data must satisfy three conditions in order to fulfill this role: (1) the data must be the right data, (2) the data must be analyzed in such a manner that the results can be easily understood, and (3) these results must be interpreted in the context of the original data (Wheeler, 1986:287). However, most texts tend to focus on the last two aspects of data.

Consequently, these more glamorous principles are also where most people trying to implement SPC begin their process. This approach leads to the problem of meaningless piles of data and predictors that are only accurate if one was lucky when designing them.

The decision regarding what data to collect should not only be the first step in SPC, but is also a very important one.

Analysis tools can indicate the areas that have the most potential to improve the process, but if the capability of the process is not measured, time may be wasted collecting data that is meaningless. Therefore, before metrics are collected and analyzed the stability of the process must be determined. If the process does not display control, it will be impossible to implement any productive change based upon metrics collected.

There are three tools that can be used to determine what data is significant to collect: the Cause and Effect Diagram, the Pareto Diagram, and the Kendall Concordance Coefficient.

3.4.2 Cause and Effect Diagram. Often there are a large number of factors involved in a problem. The Cause and Effect Diagram provides an organized approach to establish statistical control. The versatility of Cause and Effect Diagrams comes from the easy method by which they are generated, while its power comes from its graphical representation of the relationships between problems and their sources. Wheeler lists the following steps to create a Cause and Effect Diagrams in his book Understanding Statistical Process Control:

1. Choose an effect to be studied, and write it at the end of a horizontal arrow.
2. List all the factors that influence the effect under consideration.
3. Arrange and stratify these factors. Choose the principal factors, operations, and subdivisions of activity. These form the major branches off the horizontal arrow.
4. Draw the sub-branches for the various sub-factors or sub-activities. This process of sub-dividing is continued until all variables are included on the diagram.
5. Check the diagram to make sure that all known causes of variation are included on the chart. (Wheeler, 1986:288-289)

The cause and effect diagram for this thesis, based on the current literature review, looks like:

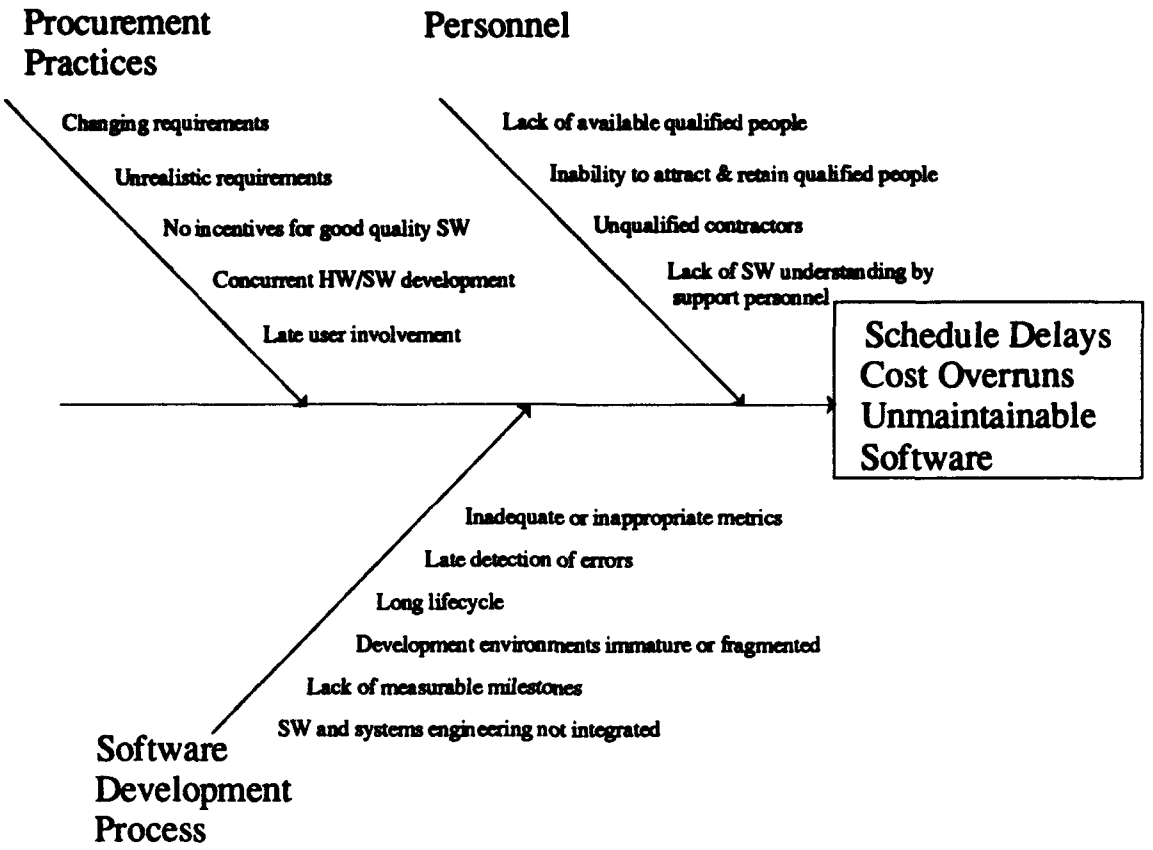


Figure 8. Software Development Problems Cause and Effect Diagram

After the Cause and Effect diagram is created it may need to be restructured in order to provide the correct relationships between all causes and their effects. Once this task is completed, each problem should be evaluated on the basis of what is currently being done to control it. Once this information is on the chart it should be obvious which areas are in need of immediate attention. Cause and Effect diagrams can be an effective way of reviewing what has already been attempted to control the process, as well as indicating the areas that will benefit most from being brought under control.

3.4.3 Pareto Charts. These charts are another simple tool that help point out the areas that offer the greatest potential for improvement. They are based on the principle that 20% of the problems will cause 80% of the headaches. Construction of the Pareto Chart consists of first deciding a factor on which to base the comparison of problems. Two frequently used factors are those of cost and incidence. Once a factor has been determined, data can be collected and placed into the appropriate category. A histogram is then created that should indicate the area with the largest problem. The main problem with Pareto Charts is they are not effective for processes that are changing over time. For this thesis it is assumed that the software development process is in the *state of chaos* and is not stable over time. Thus, Pareto Charts will not be used in this thesis.

3.4.4 Kendall's Coefficient of Concordance. Once all of the problems have been identified, it is necessary to determine from the real development projects which problems had the biggest effect on them. Once data collection is complete, a master list consisting of rankings of problems faced by different projects is assembled. The objective is to not only construct a master list, but also determine the relationship between the lists from the different projects. This analysis can be accomplished with Kendall's coefficient of concordance.

The model used for Kendall's coefficient of concordance (W) requires the following assumptions and tests the ensuing hypothesis (Daniel, 1978:327):

1. The data consists of m complete sets of observations or measurements on n objects.
2. The measurement scale is at least ordinal.
3. The observations must be capable of being converted to ranks.

H_0: The m sets of rankings are not associated.
H_1: The m sets of rankings are associated.

$$W = \frac{12\sum_{j=1}^{n} R_j^2 - 3m^2n(n+1)^2}{m^2n(n^2-1)}. \qquad (1)$$

where

W = the test statistic

n = the number of objects being ranked

m = the number of sets of rankings

R_j = the sum of the rankings for the jth object

The decision rule for determining agreement or disagreement is given by setting the critical value for W according to a required probability resulting in $W_{crit.}$ A calculated W equaling 0 indicates total disagreement, whereas a W which equals 1 indicates total agreement. Therefore, if the calculated value of W is greater than W_{crit}, the null hypothesis (H_0) is rejected, and the alternate hypothesis (H_1) will be accepted. If W is less than W_{crit}, the null hypothesis will be accepted.

3.5 Expected Results

The case study is the primary methodology used in this thesis. These case studies will provide background information on the four programs as well as detailed information on the software development process used and the software development problems encountered. The survey methodology will be used to validate the results of the case studies and to normalize the data acquired from each of the four programs.

3.5.1 Expected Case Study Results. Each case is unique and has been selected under the principle of replication mentioned earlier so that they either produce similar results, or produce contrary results for predictable reasons. Projects that have had a bad software history as well as those that have done well have been selected for this case study. The information gathered should provide a diverse view of the root causes of software problems.

First, a list of the most important software problems revealed by the case study methodology will be compiled. Second, the problems determined from the case studies can be used to create a cause and effect diagram for each of the programs. This diagram will provide a simple graphical view of the complex interrelationships of the software problems encountered on each program.

3.5.2 Expected Survey Results. The survey will validate the data acquired in the case studies. If the software professionals associated with each of these four programs place the same emphasis on the problems the case studies determined were significant, the case studies are validated.

Additional information will be derived from the surveys. This information will include a composite view of the significance of problems encountered across the entire population of aircraft embedded software, the measurability of these problems, and the correctability of these problems. After analyzing the data collected from the survey, we expect to have a list of problems that are rated by their importance to software development.

Kendall's Coefficient of Concordance will be used in each of the previously mentioned survey questions to determine the extent to which the software professionals agree on each question.

IV. Findings and Analysis

4.1 Introduction

This chapter is organized chronologically according to how the research was accomplished. Accordingly, minor changes in the approach to answering the thesis question are reflected as the chapter progresses. Section 4.2 will specifically address the flight control software on each of the four aircraft based on a case study methodology.

Sections 4.3 through 4.6 are in-depth case studies of each of the aircraft programs. These case studies include a background on the overall program, a description of the software process used in the program, and a summary of the software development progress to date.

Finally, Section 4.7 will summarize the information acquired in the surveys and provide an analysis of the data. Before going directly into these sections, some peculiarities of how the research was conducted through both the case study and the survey methods will be discussed.

4.1.1 Case Study. The focus of the entire thesis was originally based on the flight control software development on the four aircraft programs. This focus was based on the assumption that the flight control software should be the most common factor across all of these platforms. This would provide a common basis of comparison for how the software was developed on each program. Although the assumption that the flight control software is a common thread amongst all of these aircraft is valid, further research revealed that the process used in some of the programs to develop the flight control software was quite different from the process used to develop the remainder of the software on the program.

This discovery forced a change in the approach of the case study. We continued to research the flight control area but expanded the scope of the case study to include all of

the embedded software resident on the aircraft. This served to better analyze the policies and processes of both the program offices and the developing contractors.

Each of these case studies involved personal interviews of at least two individuals in the SPO. These individuals were all intimately familiar with the oversight of the software development by the SPO and the development processes used by the contractors. Much of the information gathered was available from GAO reports which had been generated over the past few years at the request of Congress. This was especially useful in the case of the C-17. Little information was available from the GAO from a software perspective on the F-22 since it is a relatively new program. In fact, discussions with the local GAO office indicated that such a review may be coming soon. There was conspicuously little information from the GAO on the development and modifications to the F-16. This could be due to the political nature of the multi-national program. In each case, the information in the GAO reports was found to be consistent with the personal interviews and other sources.

4.1.2 Survey. The original purpose of the survey was to support the findings of the case studies. Some difficulties were encountered in receiving an adequate response. It seems that the individuals in programs that had the most difficulty were least interested in completing the surveys. Forgoing this opportunity for assessment and process improvement may be indicative of the attitude which caused these programs to have the problems indicated in the case studies.

4.2 Flight Controls Case Studies

As this case study was developed, two things became apparent. The first item was that there are very few companies that provide flight control systems, and there are also few people who manage the acquisition of such systems for the government. The same

group of companies and people provide the flight control systems for almost all aircraft with digital flight control. The second item that became apparent was that since flight control systems are life critical systems, the process used to develop such a system has more in common with the process to develop a flight control system of a completely different aircraft than it has in common with the development of the other software on the same aircraft.

4.2.1 Background.

4.2.1.1 C-17 Flight Control System. The original premise was for the C-17 to employ a standard mechanical flight control system. However, during wind tunnel testing it was discovered that the airframe exhibited a tendency to enter a deep stall condition, and the only way to prevent this life critical condition from occurring is by limiting the angle of attack of the aircraft using a digital flight control system.

The mechanical flight control system was originally subcontracted to Honeywell. However, they were not qualified to develop the newly required quadruple redundant electronic flight control system with mechanical backup, so the SPO convinced DAC to hire General Electric (GE) to begin development of a similar system as a precautionary measure. In 1989, the Honeywell subcontract was terminated and the GE system became the primary flight control system for the C-17. At the time, it was easy for GE to jump right into the project since they were ramping down from development projects for two other aircraft flight control systems. Due to experience on the B-2 and V-22 programs, approximately 20% of the work was already completed.

The C-17 includes a mechanical reversion backup if the software fails in the primary flight control system. Although this was never intended to be demonstrated during the test program, there have been several instances of this unintentionally occurring during test flights without problems.

4.2.1.2 F-16 Flight Control System. The F-16 was originally developed using an analog flight control system. However, the analog system was replaced when the automatic terrain following capability was added to the aircraft, since the analog system was not precise enough for this capability. General Dynamics developed all of the software for the new digital flight control system, Bendix provided the hardware, and ASM provided the backup flight control software. The new system is a quad redundant single thread digital flight control system with automatic fly-up terrain following and wings leveler built in.

The program to develop the new flight control system was initiated in 1983 and consisted of three phases: Study, Development, and Flight Test. It is also important to note that the digital flight control system was *not* designed to maximize performance, but instead to emulate the analog system in operation in current F-16s. This decision provided the benefit that the analog test sequences could be used to verify the system before flight testing.

4.2.1.3 B-2 Flight Control System. The B-2 program originally planned to use a digital flight control system; however, it also has major design changes from the originally proposed system. The problem encountered on the B-2 was the combination of the 3g structural limit on the airframe, and the effects that wind gust have on the airframe with its flat wing shaped design. Instead of a dual quadruple redundant flight control system with only an α (angle of attack) feedback, the current system contains both an α and a g (gravity) feedback mechanism. Northrop has been developing the software for the flight control computer that is provided by General Electric. While other subcontractors are providing smaller parts of the system that include both hardware and software.

4.2.1.4 F-22 Flight Control System. The flight control system of the F-22 prototype in the Dem/Val phase was a Triplex digital flight control system. However, since this system was not developed under the same rigid standards applying to the EMD

contract of the F-22, as well as airframe modifications, a new quadruple redundant flight control system is being developed during the EMD phase of development.

4.2.2 Software Process. In analyzing the four aircraft developments we found four processes that were more robust than the processes used to develop other software on the same aircraft, but also a variety in the robustness of the different aircraft flight controls. The C-17 used a process that can be likened to the one the SEI uses to describe a level one organization. Both the F-16 and the B-2 use a process that can be likened to a level two organization, while the F-22 uses a process that can be likened to that of a level three organization (Humphrey, 1989:5).

4.2.2.1 The C-17's Process. The software development process used by General Electric is assumed to be the least robust of the four programs because of the problems they have encountered. This is presumed only because the SPO has been given very little access to the actual development process by the prime contractor, Douglas Aircraft Corporation. GE previously developed flight control hardware, but the C-17's flight control system is the first one they have provided the software for. From the limited access the SPO has been given, the flight control software is being developed a little more robustly than the rest of the software, but is also behind schedule and over budget. This fact has led to shortcuts in the development process – most typical being the deferment of documentation until after all the coding is finished.

The process used to develop the C-17's digital flight control system can be best compared to that used to develop the avionics software on the F-16 (see section 4.5.2). It is not a very structured process, and an attempt is made at the end to inspect quality into the product, instead of designing it into the product in the first place.

4.2.2.2 The F-16's and the B-2's Process. The fundamental difference in the process used to develop the digital flight control systems for these two aircraft in comparison to the C-17's is that on these programs an attempt has been made to design

quality into the software instead of only performing testing after it is developed. Both of these programs employ the use of peer reviews during the design and coding phases to try and eliminate some of their problems earlier in the development cycle. Independent verification and validation was also used on both flight control projects. Control boards were used to maintain developmental configuration control during the development process. Even though both of these programs used these techniques to try to eliminate errors earlier in the process, they also used a more robust testing program than the C-17. For all integration tests on both projects, not only are tests performed to evaluate the success of the newly added units, but full regression testing is also completed.

The process used on both of these aircraft's flight control systems is better than that used for avionics on any of the aircraft studied, except for the F-22. The improved process appears to be worthwhile, since the F-16 flight control system has had relatively few modifications since its development.

4.2.2.3 The F-22's Process. The process used to develop the flight control system for the F-22 is similar to that used on the B-2 and the F-16. However, in addition to completing the items mention for those programs, the F-22 adds some additional features that make its process even more robust. The key difference is that management is focusing on the process used to develop the flight control system on the F-22 instead of the product itself. Automation of many of the procedures like regression testing is another item the F-22 team is accomplishing. The F-22 has also formalized the team relationship that was found in both the F-16 and B-2 flight control areas, but not necessarily in their avionics areas. The F-22 has what appears to be the best process to develop software for flight control systems of the four programs reviewed, and it is believed that this improved process will also provide the best product, but only time will tell.

4.3 C-17 Case Study

4.3.1 Background. The C-17 was programmed to replace the aging C-5 fleet at a reduced maintenance and operating cost and provide the capability of using short, unimproved runways and hauling large loads over long distances in the same aircraft. These requirements were to be accomplished by integrating existing technology into an aircraft which only required a three person crew (pilot, copilot and loadmaster).

Through FY93, Congress has appropriated $13.3 billion for the C-17 program including $5.4 billion for research, development, test and evaluation; $7.8 billion for procurement; and $149 million for military construction (GAO, 1993-6:2). In April, 1990, as a result of a Major Aircraft Review, the Secretary of Defense reduced the number of planned production aircraft from 210 to 120 at an estimated total cost of $35.8 billion (GAO, 1992-205:1). The full-rate-production decision is planned for 1995.

4.3.1.1 Contract. The C-17 contract was awarded to Douglas Aircraft Corporation (DAC) in July, 1982. The contract was competed as fixed-price incentive fee (FPIF) for two reasons. First, the mandate of Congress and the current administration was to eliminate cost-plus contracts in order to reduce the excessive overruns of the past. Second, the C-17 was supposed to be an integration of commercial off-the-shelf (COTS) hardware and software, indicating low risk to the contractor. Douglas was given Total System Responsibility for the overall specified performance of the aircraft and meeting schedule milestones.

In addition to the test aircraft and two non-flying test airframes the original contract included two production options for a total of six aircraft. The ceiling price for the development contract along with the six aircraft is $6.637 billion (GAO, 1992-205:2). A separate fixed-price incentive fee contract was awarded in July, 1991 for four additional production aircraft for a ceiling price of $1.2 billion (GAO, 1993-6:2). Finally, the Air Force has awarded over $1 billion in undefinitized long-lead contracts for 18 aircraft.

Since Mil Std 2167 and Mil Std 2168 had not been written when the initial contract was competed, only the following standards apply to the C-17 contract:

Mil Std 483 (before Mil Std 2167 or 2168) - Configuration Management for Systems, Equipment, Munitions, and Computer Programs
Mil Std 1521 - Technical Reviews and Audits for Systems, Equipments, and Computer Software
Mil Std 490 - Specification Practices
Older DIDs

4.3.1.2 Program History. The C-17 development has suffered significant delays due, in part, to problems associated with poor management of the contract and underestimation of the scope of the effort required. DAC's major management deficiency was their lack of domain knowledge. Most of the personnel working at DAC had never been involved with a military aircraft development before. Also, both DAC and the SPO underestimated the scope of the effort necessary to meet the requirement for a three person crew. Consequently, the original assumption that the C-17 was simply an integration of COTS avionics was incorrect. The three person concept, along with problems associated with the flight characteristics of the aircraft, generated the need to digitize many of the subsystems.

The program office was limited in its ability to monitor the contract due to the Paperwork Reduction Act. This act was an attempt at reducing the burden on the contractors for the seemingly meaningless piles of paperwork associated with DoD contracts. However, this eliminated key documents required by the program office to track the contractor's progress. This act also reduced the insight of the program office into DAC's development process and adherence to quality standards.

Although the ceiling price was $6.637 billion for the original contract, the government estimates that the completion cost will be over $7.5 billion. Of course, since this is a fixed-price contract, the contractor will bear the additional cost. Cost performance data in

1992 reveals that Douglas has only earned $0.69 for $1.00 of planned work on the development contract (GAO, 1992-205:2).

4.3.1.3 Current Status. The C-17 program is currently conducting Developmental Test, and Evaluation (DT&E), with limited rate production approved. DAC has delivered the 3 aircraft required for DT&E, and 4 of the 6 production aircraft required for LRIP. All of these aircraft have been accepted with open items since many of the specifications have not yet been approved. DAC retains retrofit responsibility for all of these items as well as any discrepancies found during the test program.

The software on the first test aircraft was supposed to support all avionics functions. However, the delay in the software development program has resulted in planning for full functionality in the first production aircraft. This goal has repeatedly slipped and software development has been postponed rather than slow aircraft production (GAO, 1993-6:41).

4.3.1.4 Subcontracts. The majority of the software for the C-17 has been subcontracted by DAC. In fact, over 3,500 subcontractors or suppliers have been identified, of which 33 are considered critical (GAO, 1992-207). The mission computer and weight-and-balance software are the only CPCIs done in house. The other 56 Computer Program Configuration Items (CPCI) were subcontracted to 15 subcontractors. During this process DAC did not specify a language to use and thus the C-17 has software in almost every known language of that time. Because only 4 of the final 58 CPCIs were on the original contract, the procurement specifications between DAC and the subcontractors were not subject to AF approval. However, through AF pressure the documentation for the additional CPCIs was included in the contract at no cost.

Management of the subcontractors by DAC has also proved to be a problem. A major portion of the software problems have occurred on the mission computer software and flight control software. Both of these have fallen behind schedule, delaying first flight and resulting in reduced functionality in the initial production aircraft. The SPO reported in

1988 that the mission computer delays were caused by a delay in awarding the subcontract, inadequate definition of requirements, underestimation of the work required, and overestimation of software engineer productivity (GAO, 1989-195:18).

4.3.2 Software Process. "The C-17 will be the most computerized, software-intensive aircraft ever built, relying on 19 different embedded computers incorporating more than 80 microprocessors and about 1.3 million lines of code" (GAO, 1993-6:41). This quote in itself explains why the software process is particularly important. Unfortunately, the process was not well-defined at the start of FSD and is still not well-defined.

4.3.2.1 Software Process Model. Although the software on the program was produced by 15 different subcontractors, the waterfall process model was the primary model used at the beginning of the development. However, because of the combination of schedule delays and the risk of termination if certain milestones were not met, the waterfall process degenerated into an incremental process. The interim capabilities provided by the incremental development satisfied the need to demonstrate progress and pacify higher government officials.

4.3.2.2 Ensuring a Quality Software Product. DAC has a software quality assurance organization which does inspections and final review of testing documentation – basically the standard minimum practice for defense contractors. The quality assurance specifications for many of the CPCIs have yet to be agreed upon between DAC and the SPO. Therefore, the current level of quality assurance is that specified by the applicable MIL Standards and the individual company quality assurance policies.

Although JOVIAL was the preferred programming language by the DoD at the time, no requirement was stated in the contract for its use. As a result, almost every known programming language is now represented on the C-17 platform. This variety of languages is obviously a significant impact to maintainability and understandability in the future.

Once again, the Paperwork Reduction Act hindered the SPOs ability to properly monitor DAC and ensure the quality of the software produced. In addition, a strained working relationship between DAC and the SPO further reduced communications and the ability of the program office to work closely with the subcontractors. Although the SPO was invited to major reviews at the subcontractors, further insight into the quality of the work accomplished at the subcontract level was very limited. In some cases only the threat of disapproval of required documentation enabled the SPO to communicate more freely with the subcontractors.

Because of the program office's mistaken assumption that the software development would be trivial, Mil Std 52779A, which requires the contractor to establish a software quality assurance program, was not placed on the FSD contract. As a result, DAC was not required to establish the normal type of software quality assurance program seen in most weapon systems. Although DAC had several people responsible for software quality assurance, both the DPRO and the DoD Inspector General reported that their approach was inadequate and undisciplined (GAO, 1992-48:22). These reports pointed out that software quality assurance was understaffed and had no significant management authority to correct problems.

At the SPO's encouragement, DAC finally agreed to comply with a modified version of the software quality assurance standard in 1988. However, in late 1990 the DPRO reported that the software quality assurance program still suffered from the lack of independence and management authority needed, citing that upper management often ignored their recommendations (GAO, 1992-48:22). Although this problem is supposedly now corrected, a large portion of the software development has already occurred resulting in overall poor quality software.

Poor quality is evident in the Defense Director of Operational Test and Evaluation statement that software documentation was inadequate and that the Air Force's ability to

maintain the software may be impaired (GAO, 1993-2:4). Less than half of the avionics subsystems which the Air Force plans to maintain have approved documentation (GAO, 1993-2:8). This means that the Air Force will be forced to rely heavily on DAC and the subcontractors for software support.

4.3.2.3 Software Process Improvement Efforts. DAC has absolutely no software improvement process. Basically, it is all that they can do to keep their head above water since they are considerably behind schedule and above cost ($1.2 to $1.7 billion) on a fixed price contract. There is no support from upper management to spend any money on process improvement in the short term to get long term improvement – even for significant future cost savings. Upper management's current philosophy seems to be to slug their way through completion of the full-scale development contract in order to recoup their losses once the production contract is awarded. In fact, DAC has actually reduced the number of personnel working on the program to reduce short-term costs.

DAC has never had an SEI assessment and, because of its short-sighted philosophy, is not likely to have one in the near future. They have no other known plans for implementing a process improvement program.

4.3.2.4 Oversight by SPO. The SPO has a significant cadre throughout many disciplines dedicated to monitoring the software process. Approximately 25 out of the 300 people in the SPO are specifically dedicated to software. There are 6 computer resources acquisition managers, 13 to 15 software engineers and software specialists in manufacturing and logistics.

Most of the time the SPO communicates directly with DAC, and only rarely deals with the subcontractors. The SPO is currently in a monitoring mode due, in part, to the fact that this is a firm fixed-price contract and that the contractor has total system responsibility. However, the SPO does participate in major reviews such as PDRs, CDRs, and TRRs. Refinement and approval of specifications and other documentation are the

primary software activity of the SPO when dealing with the contractor. The SPO rarely reviews the actual work completed by the subcontractors. The only time that the engineers actually look at the code is at the TRRs – and then only samples.

The C-17 has repeatedly been cited by the GAO as being a good example of how *not* to manage software development on a weapon system. The Air Force initially saw the software development as being low risk and therefore did little to either manage it or monitor the contractor's performance.

A 1992 GAO report summarized the mistakes which affected the Air Force's ability to oversee the software development. According to this report, the Air Force

- underestimated the size and complexity of the software development effort;
- assumed that C-17 software development would be low-risk without performing the type of analysis necessary to support and document that assumption;
- either waived or ignored many of the DoD standards and guidance for managing software development, despite DAC's limited software development and integration experience; and
- awarded a contract that (1) gave Douglas the control over software development, (2) limited the Air Force's access to software cost, schedule, and performance information; and (3) restricted the Air Force's ability to require corrective actions, even when critical software problems became evident. (GAO, 1992-48:4)

4.3.3 Software Development Progress. The GAO identified three major areas where software development continues to be a problem (GAO, 1993-2:2). First, critical software functions are still in either development or test. Second, reserve processing and memory capacity requirements for future growth are not being met. Third, system documentation necessary to efficiently test, maintain, and upgrade the C-17 computer systems has not been completed.

The Defense Director of Operational Test and Evaluation, following completion of an Early Operational Assessment of the C-17 in December 1992, concluded that immature software has limited testing of the C-17's operational capabilities such as flight controls, stall warning, communications, aerial delivery, and navigation (GAO, 1993-2:4).

Operational testing planned for September 1993 have now been delayed until January 1994 due to these shortfalls.

4.3.3.1 Software Problems Encountered. The program schedule is so far behind that the only incentive for DAC to complete the FSD contract is to get to the production contract. The late software schedule has resulted in the necessity of incremental development, progressing from basic functionality to full functionality, due to the threat of cancellation by Congress if progress is not demonstrated. This change in software development strategy is highlighted in a 1989 GAO report, on the schedule risks associated with the transition to production (GAO, 1989-195:18-19). Some possible root causes of the late software include:

Requirements changes. As in any weapon system development, changes in requirements by the user have had an impact on the schedule. However, GAO reports from both 1989 and 1992 indicated that some requirements were actually reduced for the test aircraft.

Inadequate Requirements Definition. This was identified by the SPO as a major delay in the mission computer software.

Underestimation of Work Required. Software size and complexity estimation has been historically difficult on most weapon systems. This underestimation caused inadequate personnel resources to be allocated and resulted in significant schedule delays. Software risks were underestimated and not managed by either DAC or the Air Force. As a result, the schedule was far behind before any problems were recognized.

Overestimation of Software Engineer Productivity. This serves to compound the problem of meeting schedule. If productivity is overestimated, fewer resources are allocated to the task. This may have been avoided by a software process assessment of the prime contractor and the major subcontractors.

Lack of domain knowledge. DAC has not built an aircraft in a long time, their personnel are inexperienced in the process of developing the systems required for a major aircraft.

Progression to digital implementation on many subsystems. Due to the requirement that the aircraft be manned by only three crew members, DAC needed to digitize many of the avionics and other subsystems that require manual operation. This process was a major reason that the current aircraft has 58 CPCIs compared to the four that were originally proposed.

Paperwork reduction act. The reduced oversight caused by this act resulted in the delay of the discovery of errors and misunderstandings until further in the process. Historically, the longer errors remain undetected, the more time and money it costs to correct them.

CPDSs still not approved. DAC and the government have only been able to agree on section III, detailed requirements, of the Computer Program Design Specifications (CPDS). The sections on scope, applicable documents, quality assurance, and preparation for delivery and shipment have not been finalized.

Documentation not satisfactory to date. Much of the documentation that has been presented to the SPO for approval has not been satisfactory. This has caused delays due to reworking the documentation and the inability to progress until approval has been received.

Personnel reduction to reduce costs. In an effort to reduce short term costs and limit the amount of money that they will lose on this contract, DAC has reduced staffing on the program. This has caused not only a stretch-out of the software development effort, but also removed people that were familiar with the various software units. This has increased the time to complete units due to the extra time needed by the eventual programmer to understand the code they did not write.

Digital flight controls. In order to compensate for a deep stall condition the addition of a digital flight control system was essential. The change to a digital flight control system created the need to digitize many subsystems on which they had not planned. This also precipitated another major problem – the subcontractor developing the initial flight control system was not technically capable of delivering a digital flight control system and had to be replaced midstream.

4.3.4 Program Office Actions Taken. The SPO has been very limited in what action they could take due to the FFP contract. However, they have used their power of approval over specifications and test results to monitor DAC and force them to modify items done incorrectly. The AF also has a warranty on the C-17 that the SPO can use to force the contractor to correct most deficiencies.

In most cases the SPO has tried to work with the contractor in order to get the aircraft delivered in a timely fashion. They approved the request for incremental development of some of the software, allowing the contractor to achieve certain milestones at an earlier time.

The SPO has also relaxed some of the non critical requirements which the contractor was close to meeting. Compliance with these requirements would have created a considerable cost burden on the contractor.

The AF has not required all specification items to be closed before accepting an aircraft, allowing the contractor to correct these deficiencies at a later time. Certain requirements were deleted entirely for the first test aircraft in order to begin testing.

4.3.5 Case Study Summary. The problems associated with the C-17 software development are evenly spread amongst the problem categories mentioned in the cause and effect diagram in chapter three — procurement practices, software development process, and personnel. Contributing problems under procurement practices include a lack of management priority toward software, unrealistic schedules/manpower estimates,

undisciplined process to baseline requirements, and management's focus on hardware vs software. Software development problems include inadequate software development planning, lack of standardized software development techniques, immaturity of the contractor development process, and insufficient documentation. Personnel problems include lack of experience of both the contractor software engineers and management. Depending on your point of view, you could also cite incompetence of government personnel as a problem in this category.

Therefore, in the case of the C-17, there is a wide diversity of problems in the different categories. Although the immaturity of the contractor's software development process and the low management priority toward software are the most significant overall factors, there is clearly no easy fix to the problems mentioned above. SPO relief of contractor schedule and quality requirements can also be seen as a significant factor since the contractor had little incentive to fix its process. However, metrics used in each of these areas would allow both the contractor and SPO to effectively manage these risk areas and measure improvement when it occurs.

Given the specific problems mentioned in this case study, a more appropriate cause and effect diagram for the C-17 SPO is found on the following page:

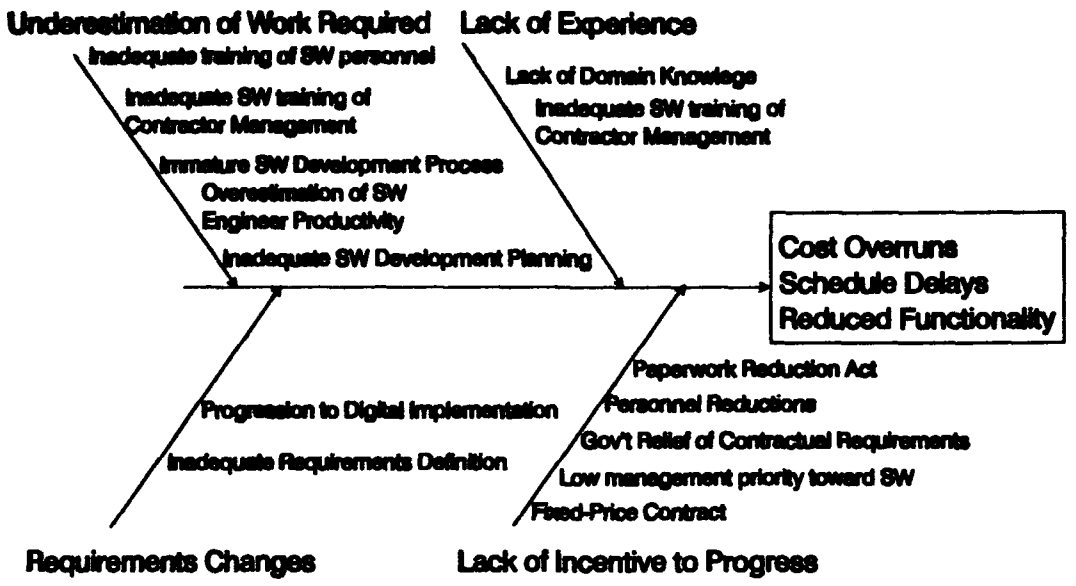


Figure 9. C-17 Cause and Effect Diagram

4.4 The F-16 Case Study

4.4.1 Background. The Mideast war of 1973 alerted US tactical planners that numbers of fighter aircraft are as critical as performance in battles to win and maintain air superiority ("War Spurred...," 1977:71). The F-16 was developed to fill this gap. It was designed to serve the air superiority mission as well as having the characteristics to fly intercept, interdiction and close air support missions. The critical design goal of the Air Force was an aircraft that could outmaneuver Soviet fighters at altitudes up to 40,000 ft at Mach numbers of 0.6 to 1.6.

4.4.1.1 Contract. The F-16 Full Scale Development contract was awarded to General Dynamics (now Lockheed, Fort Worth) – in 1975. As with most of the weapon systems under development at this time the contract was firm fixed price. An incentive fee was added based on the target unit production cost of $4.55M in 1975 dollars (Kolcum, 1977:47).

This was a complex development which involved five NATO allies – Belgium, Holland, Denmark, Norway, and the United States. The Memorandum of Understanding (MOU) between the U.S. and these other countries clearly delineated which portions of the aircraft would be developed and produced by subcontractors in each country. Included in this MOU was a not-to-exceed price of $6.091 million per aircraft (Ropelewski, 1977:59). Originally, 1,636 aircraft were planned for production amongst the five countries.

4.4.1.2 Program History. The F-16 evolved from the lightweight fighter program which was initiated in 1970 when deputy Defense Secretary David Packard "directed the Air Force and the Navy to nominate innovative program for prototyping with the object of reversing the mushrooming cost of buying weapons" ("War Spurred...," 1977:71). In mid-1971 the lightweight prototype project was approved for approximately $100 million. Of this $100 million, General Dynamics was awarded a $38 million contract to develop and produce two YF-16s and Northrop was awarded a $39 million contract to develop and produce two YF-17s ("War Spurred...," 1977:71). The remainder of the money went to General Electric and Pratt & Whitney for engine development.

The Dem/Val phase was concluded by a fly-off between the two aircraft. Although both aircraft were acceptable to the European countries at the conclusion of Dem/Val, the YF-16 was selected by the U.S. Air Force based on marginal performance and cost considerations. The YF-16 was also considered to be closer to the production version than the YF-17 ("War Spurred...," 1977:72).

The Full Scale Development contract was awarded in 1975 to General Dynamics with an anticipated production of 650 planes for the U.S. and 350 planes for the four European nations ("Defense Contract: ...," 1975:70). On 7 De[illegible]77, Deputy Secretary of Defense C. W. Duncan made the production decision in which 1,388 USAF and 348 European aircraft were projected (Geiger, 1977:xxiii). The USAF production of the F-16 started in

May 1978 with the execution of the first production option on the FSD contract for 105 aircraft. The second production option was exercised in November 1978 for 145 aircraft (Geiger, 1981:29).

To date, over 3,500 F-16s have been produced for over 20 countries around the world (Geiger, 1987:167). The design of the aircraft has evolved through seven major block changes to perform multiple air combat roles. The most recent information from the F-16 program office indicates that USAF production of the F-16 block 50 aircraft will cease in FY94. However, continuing modifications and foreign military sales will continue to be major activities far into the foreseeable future.

4.4.1.3 Current Status. The F-16A and F-16B models have been PMRT'd to Ogden Air Logistic Center where all software modifications/maintenance are accomplished. The F-16C and F-16D models are still managed at the program office at the Aeronautical System Center. New F-16C/D models are still being produced for the U.S. Air Force through 1994, while aircraft that have already been fielded are being modified. The SPO is currently managing 38 different programs, some of which consist of new aircraft procurement for FMS customers, while others consist of modification efforts.

4.4.2 Software Process.

4.4.2.1 Software Process Model. The original software development effort occurred before software process models were developed. Therefore, the code and fix process model (no model used) was the software development process model by default. Most of the code in the original versions of the F-16 consisted of Jovial spaghetti code. To this date, most of the software development on the F-16 has used the code and fix process. Therefore, the code going into current production versions of the F-16 is also mostly Jovial spaghetti code.

The software personnel in the SPO realize that this is not a very good way to do business and are trying to implement changes. On the latest major modification, the

Modular Mission Computer (MMC), the SPO has dictated the use of Ada, the waterfall process model, and object oriented analysis and design.

4.4.2.2 Ensuring a Quality Software Product. The SPO has not really attempted to ensure a quality product over the years. Currently there are only five people in the SPO that possess any training in software development management. These individuals are doing everything they can to improve the quality of new software development projects. However, up to the current time, no effort has been made to place new DoD software standards, new software engineering techniques, or other initiatives that could improve the quality of the software. Instead, the SPO took whatever poor documentation and unmaintainable software General Dynamics delivered. The philosophy seemed to be that schedule was a more important factor than cost – plenty of money was available as long as the program was kept on schedule. This gave the contractor little incentive to improve efficiency and productivity by using new software engineering methods. The affluence of money and the importance of the schedule led to other peculiarities in software acquisition.

Rather than generating most software modifications by ECPs, the F-16 SPO generates a prospective list of modifications through solicitations from the users. This list is then presented to a board of operators who determine a rank order list of the prospective changes. This list is then presented to the contractor with a schedule and a fixed amount of money. General Dynamics then informs the Air Force how many of the top problems they will attempt to address. In effect, this wish list is simply an organized system of advocating requirements creep. Since no statement of work is prepared, specifications of any type are not sent to the contractor, and individual changes are not priced out, the contractor is basically operating on a time and materials basis on requirements which are barely documented.

Since the Air Force does no research into the size of these changes, they have not attempted to implement Ada or any new process model during implementation of the modifications. All of these changes are considered to be minor no matter how much code eventually changes; therefore, the code and fix process model is used to make these changes. Since Ada was mandated in 1987 by DoD Directive 3405.2, *Use of Ada in Weapon Systems*, not a single waiver has been submitted on the F-16 program for any software changes placed on the wish lists, even though modifications performed by the contractor ended up changing more than the 1/3 of the code allowed in the directive (Collins, 1993).

Another problem is that these changes are not tested at the unit level. Instead each change is added to the B-5 specification which is maintained by the contractor and is not subject to Air Force approval. There is a large variation in the magnitude of the items on these wish lists. However, even the insignificant items that should be changes to the C-level specifications, as well as items that belong in the system level specification, are being recorded in the B-5 specification.

When the schedule dictates that testing is to begin, a complete regression test is then performed to determine that the system meets all of the items on the B-5 specification. After all of the B-5 specification items are satisfied, the code is uploaded onto a simulator where it is then stress tested by independent operationally-oriented testers. Any new problems discovered at this point are deferred to the list of software items that need to be fixed next time.

4.4.2.3 Software Process Improvement Efforts.

Software quality assurance was not an original requirement on the F-16 development contract. Although the Government always relied on General Dynamics to manage the details of the software development, this oversight was eventually corrected by a modification to the contract to include a formal software quality assurance activity.

Accordingly, 9.6% of the cost of the engineering activity was added to each system modification to accomplish software quality assurance. However, it was later discovered that there was no tasking associated with this additional cost and little was being done by the contractor to assure software quality.

This gradual recognition by the government that software quality assurance was a problem at General Dynamics was eventually corrected by an increased focus by upper management. This resulted in the assignment of additional personnel from the quality assurance area to the F-16 program and eventually led to the co-location of software quality assurance personnel with the software developers. In more recent modifications to the software, Mil Std 2168 is included to ensure a more formal software quality program.

Another quality issue involves the establishment and influence of the Software Engineering Process Group (SEPG). The SEPG was originally established as a division-level organization to establish and change general software development process guidelines. Since the SEPG was not originally directly affiliated with the F-16 program and it had little authority from upper management, mid-level and low-level management on the F-16 program typically ignored any policy or process changes which were recommended. However, with the increased Government attention to the software quality requirements in the contract, upper management gave the SEPG more authority to enforce its policy changes.

As a result, the F-16 program is conforming more to the software development process used throughout the rest of the company. The SEPG now has adequate representation from the F-16 program. The most significant accomplishment of the SEPG to date is the incorporation of Mil Std 2167A and Mil Std 1803 along with 21 pages of tailoring into the software development process used on the F-16 program.

Many of these changes in the F-16 software development process have come about as the result of pressure exerted by the SPO to modernize software development practices to

those of current weapon systems. Until recently, General Dynamics wrote software very much the same as they did in the 1970s. The immaturity of the F-16 software development process has been a well-guarded secret by General Dynamics. In fact, they were proud to show that an SEI assessment of the division showed them to be a level 2 organization. However, they may fail to mention that the F-16 program as a whole was not included in this assessment – the F-22, Modular Mission Computer, and two radar programs were evaluated.

Over the past two years the SPO has attempted to improve the efficiency of the contractor's software development process. This has been accomplished through cooperation with the contractor's SEPG and enforcing adherence to standards which have been initiated in new engineering change proposals.

Ensuring quality software is a virtually impossible task for the SPO since there are fewer than five individuals in the SPO with any significant software experience. Instead, the new philosophy by the SPO is to ensure that the contractor's software development process is sound and that software quality planning is taking place and is adhered to. To accomplish this, a Software Process Group consisting of the software experts in the SPO ensures that each new contract includes military standards 2167A (tailored), 2168, and 1803; a Systems Engineering Master Schedule (SEMS); and a Systems Engineering Management Plan (SEMP). These documents simply require the contractor to adequately define and manage the software development. In this way the SPO manages the contractors process rather than the products specifically. The assumption here is if the process is sound, the product will be good.

New software development methods such as object oriented design using Ada are also being implemented in such subsystems as the Modular Mission Computer. The major emphasis at this point is to produce cost-effective, maintainable software.

4.4.2.4 Oversight by the SPO. The F-16 SPO has very little oversight of the software development process at General Dynamics or any of the subcontractors. This lack of oversight is not because of an uncooperative prime contractor, but is due to a lack of knowledgeable software personnel in the SPO. Currently there are only two software engineers in the F-16 SPO, and an additional three people in the SPO that consider themselves somewhat capable of monitoring software development. However, there are currently 38 different projects in the SPO containing major portions of software. The software personnel in the SPO are forced to choose the critical projects to monitor and must trust the contractors to deliver quality software on the others. Important systems like the flight control computer do not have a high enough priority to rate SPO oversight. To say the least, this arrangement is very unnerving.

4.4.3 Software Development Progress. The lack of government oversight into the inner workings of the software development results in little to report on the software development. The F-16 suffers from the legacy of the programming practices used in the 1970s. Most of the activity over the past decade has involved changing the original code generated for the early production aircraft. Software experts in the SPO regard the software process in past years as schedule driven with little emphasis on quality, efficiency, or cost. However, little quantitative information is available to support this claim.

4.4.3.1 Software Problems Encountered.

Most of the problems found on the F-16 program are due to the way the program has been managed and the lack of emphasis on cost reduction in software processes early in the program. These problems are:

Poor testing procedures. Testing procedures are grossly inefficient. The testing philosophy is also geared toward passing a test rather than finding errors.

Lack of standardized software development techniques. This problem is improving due to new SPO initiatives to include the use of software standards in new engineering changes.

Lack of adequate documentation. Little or no documentation was delivered with the code in past years. Again, this is being corrected by the enforcement of military standards.

Inadequate software development planning. This has not been a contractual requirement until recently.

Lack of SPO oversight. Lack of software personnel resources allows the software to be managed only at a very high level.

Inadequate tracking of software costs. Neither the CPR nor the contractor's accounting records separate software costs.

High cost. This results from the problems listed above as well as the attitude in the past that schedule is more important than cost, de-emphasizing efficiency. This has also resulted from the fact that there has always been plenty of money available for modifications on the F-16 program.

Requirements creep. The F-16 program has built requirements creep into a regular process. The real requirements need to be sorted out from the users desires.

Undisciplined process to baseline requirements. The user has little oversight from DoD on how requirements are generated.

4.4.4 Case Study Summary. The problems associated with the F-16 software development are evenly balanced between procurement practices and the software development process. Problems in the procurement practices category include requirements creep, changes in user requirements, undisciplined process to baseline requirements, and lack of government oversight. Software development process problems include inadequate software development planning, use of an inappropriate process model, and lack of adequate cost metrics to measure development progress. Low management

priority toward software could also be cited for the SPO since there are so few personnel dedicated to software.

The key issue for the F-16 is the lack of government oversight. With increased government personnel the SPO could adequately monitor the contractor and force the contractor to change its development process to be more efficient. This change could result in a significant cost savings to the government in the long-run. Control of the requirements process is also a major issue which affects cost – is the Air Force processing changes on a continuing basis for real operational requirements with a cost-benefit analysis taken into account?

The following cause and effect diagram groups these problems into a more logical categories than simply procurement practices or software development process problems for the F-16 case study:

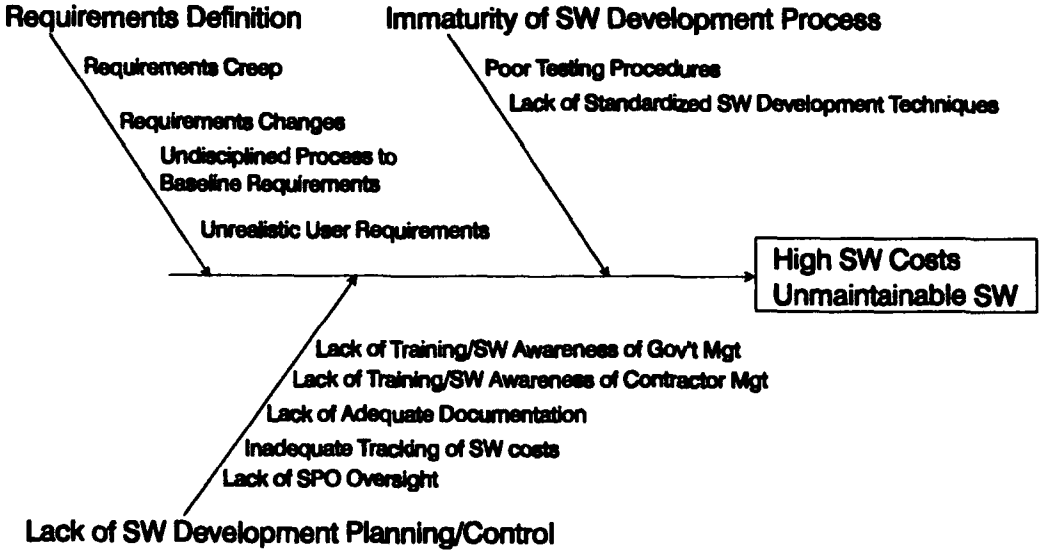


Figure 10. F-16 Cause and Effect Diagram

4.5 The B-2 Case Study

4.5.1 Background. The B-2 is designed to perform the traditional long-range bomber role for both nuclear and non-nuclear missions. It is a flying wing aircraft with two crew members and provisions for a third. In its twin weapons bays, up to 50,000 pounds total payload capacity can be carried. The B-2 design includes low observable technologies such as special shaping and radar absorbing materials, which are intended to reduce the radar cross section of the aircraft. The low observable technology combined with on-board avionics, are intended to allow penetration of current and postulated Soviet defenses.

4.5.1.1 Contract. The EMD contract for the B-2 was awarded to Northrop in the 1981. Due to the high degree of risk involved with the new technologies that were planned to be incorporated into the bomber, the contract was awarded sole source as a cost plus award fee contract. Due to the fact that the program started in the black world, it is not known if competition occurred in the earlier phases of the program or not.

Since Mil Std 2167 and 2168 had not been written when the contract was awarded, the following software standards are included in the B-2 contract:

Mil Std 483 (before Mil Std 2167 or 2168) - Configuration Management for systems, Equipment, Munitions, and Computer Programs
Mil Std 490 - Specification Practices

As the new standards were developed, the contractor chose to incorporate some of the new procedures into their software development process. This practice has been strictly voluntary and is believed to be in the best interest of both the contractor and the government.

In 1986, Northrop received authorization to begin producing aircraft. This contract is different in that out of the six developmental test aircraft, five will be converted to operational assets at the end of the flight test program, while the first unit will continue to

be used for testing throughout the life cycle. The master schedule was revised in 1989 to add two more aircraft to the development contract and delay the production decision until 1991. Since this delay occurred because of the slip in the first flight, flight testing and production still were scheduled to proceed concurrently.

4.5.1.2 Program History. The B-2 started as a black program in the late 1970s. It was originally conceived to be a high altitude bomber that would replace the aging B-52s. However, bomber penetration tactics changed around the time the program was entering EMD and the Air Force modified its requirements to add a low altitude capability. This change required a redesign of the airframe as well as adding new control surfaces. The end results were schedule slips and other problems in the EMD phase.

In January of 1986, Northrop began manufacturing the first B-2 even though the air vehicle design was not complete. This drastic step was taken to try and meet the scheduled date for first flight. However, by starting manufacture before the aircraft design had stabilized, manufacturing personnel were receiving engineering drawings late or were not able to use the drawings they received, and were forced to wait for new drawings and parts (GAO, 1990-284:13). The first flight of the B-2 occurred 19 months after originally scheduled despite the extra cost and effort allocated to try and save the schedule.

In April of 1990, Department of Defense Secretary Dick Cheney made the decision to reduce the total number of B-2s to be acquired from 132 to 75. This reduction was due to a revision in the U.S. strategic targeting plan that occurred in reaction to the collapse of the Soviet Union (Morrocco, 1990:18). Changing the number of aircraft procured reduced the total program costs from $75.4 billion to $61.1 billion; however, it increased the cost per aircraft from roughly $570 million to $815 million according to 1990 estimates (Morrocco, 1990: 19).

Rep Les Aspin (D.-WI), chairman of the House Armed Services Committee, felt secretary Cheney's proposal moved in the right direction, but not far enough. In 1990, the

House Armed Services Committee voted to cancel production of the B-2 Aircraft. However, the Senate voted to continue the planned production. The B-2 program was preserved during reconciliation meetings mostly due to the strong support of Senator Sam Nunn (D.-GA). He pointed out,

> four B-2s could have carried out the raid over Libya that required two carrier battle groups, prepositioned assets, foreign bases, and Air Force planes for a total of 84 combat aircraft and 35 support aircraft. Over Libya, 134 aircrew members were at risk. Four B-2s and four KC-10 tankers operating from American Bases would have placed only eight crewmen at risk at a cost of $4 billion. Two carrier task forces cost $30 billion. The B-2s could have accomplished the raid over Libya in hours, not days. In short, the B-2 was not expensive when one considered the cost of alternatives. (Wolf, 1990:322)

Even with the Senator's convincing argument, Congress continues to debate the fate of the B-2 production program every year. It appears that instead of the 75 requested by the Air Force, congress will stop production of the B-2 at the 20 aircraft already authorized, or in other words, one squadron of B-2s to be located at Whiteman AFB, MO.

4.5.1.3 Current Status. The B-2 program is now concurrently developing and producing aircraft. The production phase is ramping up and is scheduled to deliver the first production airplane in December of 1993. Multiple aircraft have already been delivered for flight testing. These aircraft do not possess all of the functionality that will be incorporated into the final versions of the B-2 since the EMD part of the concurrent development process is not yet complete.

The EMD team is developing the aircraft with an evolutionary process model. The first iteration included all of the functions needed to start flight tests, future iterations are gradually increasing the functionality of the Weapon System. Both hardware and software are being developed to accomplish these needed functions. The combination of the concurrent development, testing, and production, not to mention the evolutionary nature

of the process has created a configuration control nightmare. However, it appears that both the SPO and Northrop are doing an excellent job of managing it.

4.5.2 Software Process.

4.5.2.1 Software Process Model. As mentioned earlier, the B-2 is being developed using an incremental or evolutionary process model. The big picture is to have three separate increments, the first build consisting of all of the functions needed to start flight testing, the second build consisting of the functions needed to give it minimum operational capability, and the third build completing the functionality of the aircraft. Each of these three builds is composed of 10-20 major functions.

4.5.2.2 Ensuring Quality Software. The SPO and the contractor are working together as a team to try and develop the software development process being used to create the software for the B-2. Most of the problems encountered on the program were caused by management and technological failures. Other problems have been encountered, but they were not very significant when compared to the major problems in the program. However, the fact that the SPO and the contractor are working together, along with their commitment to process improvement, is minimizing the effects of these smaller software problems.

In an effort to monitor the progress of the software development and minimize the risk associated with it, the SPO and the contractor have four short meetings everyday to evaluate problems. The first meeting of the day, is a standup to brief any software problem reports (SPR) that have been closed out any new software problems that were discovered the previous day. At this meeting someone is given the responsibility to investigate all new problems that have been presented. At noon, the investigators assigned at the morning meeting, present their findings and the problems are either made into an SPR or consigned to more study. After this second meeting, the new SPRs are entered into the data base, and charts are prepared to show the status of all the SPRs. A third

meeting is then held to assign an individual responsibility for correction of all new SPRs. They also review the charts produced from the database. The fourth meeting is then held with the B-2 project leader. In this meeting he is briefed on all important aspects of the software development effort.

Another method being used to ensure quality software is the development and testing method employed on the B-2. Peer-review walk-throughs are conducted on each module during development. Each unit is first tested as an individual unit, and after passing these tests is introduced to with the current integrated software build. At this point both integration tests specific to that unit and regression tests are performed. If no problems are encountered, the new build is then loaded into the "Iron Bird" simulation computer. The Iron Bird is a mock-up of the B-2 that does not fly but tests all of the flight control characteristics. Regression tests are then performed to make sure that the new build does not interfere with any flight control issues. The software is then flight tested on the advanced flying test bed, a modified KC-135. If no problems have been encountered any time in the test process, the software is loaded into one of the six test B-2 aircraft and flight tests are performed. If a problem is encountered anywhere in the process, an SPR is written and the unit is returned to the programmer for correction after it has been removed from all of the integrated builds.

As they improve their software development process, they are implementing new procedures to ensure a quality product is produced.

4.5.2.3 Software Process Improvement. Northrop has completed two self evaluations, using a team that was trained at the Software Engineering Institute (SEI). The first evaluation placed them at level 1, and they were able to improve to a level two for the second evaluation. Both the SPO and Northrop are working together to improve the software process.

Northrop has turned to process improvement out of necessity, not by choice. Their organization was neither equipped to deal with a software development project of the magnitude of the B-2, nor did they realize this fact at the beginning of the program. After they started working on the program, this fact became apparent. At this time a decision was made by Northrop to embrace the principals of the SEI's improvement process. This decision was supported by the Air Force. Together, Northrop and the Air Force have worked to strengthen Northrop's software development capability. Their first evaluation placed their organization at a low level 1 and focused their improvement efforts on developing standardized procedures.

Nine months later Northrop performed a second evaluation on themselves and gave themselves a rating of high level 2. It is questionable whether an organization can go from a low level one organization to a high level 2 organization in nine months — the SEI recommends two years between evaluations. However, the fact that they are effectively managing configuration control of their difficult evolutionary design process, it may be true.

4.5.2.4 Oversight by SPO. The B-2 SPO is organized using the integrated process team (IPT) structure. This structure consists of teams that are responsible for all parts of the aircraft. The top level team consists of the actual B-2. It is co-chaired by the SPO program director and the program manager from Northrop. The members of the team consist of all of the functional heads from both the SPO and the contractor. The next level down consists of three teams: one for EMD, one for Production, and one for Operational Deployment. Once again these teams consist of the key members from both the SPO and the prime contractor. Each team is also co-chaired by a member from each of these two components. This structure continues down to the lowest levels of the program. It places the members of the SPO and the contractor together for all of the

important decisions and provides the SPO with excellent insight into the development process.

4.5.3 Software Development Progress. The B-2 software is far beyond the initial estimates for both cost and schedule. Software development for originally planned functions will probably still be going on after the majority of the aircraft have been delivered to the user. Even though this situation sounds very discouraging, it is actually not that severe. The B-2 is a prime example of the cutting edge of technology, and how pre-planned product improvement can be used to harness rapidly developing technology. Software development on the B-2 has both benefited and suffered from this cutting edge of technology. Most of the advanced developments are occurring in the hardware development, but with the new hardware brings new software requirements. These new requirements have fit neatly into the incremental software development process, and help justify longer schedules. However, the magnitude of the software development task for the B-2 was greatly underestimated at the start of the project, and they are still trying to catch up. Each new requirement forces a re-evaluation of the impacted design structure and shifts productivity from forward progress to rework. The managers in the SPO and at Northrop have a much greater understanding of the magnitude of the task facing them today and are effectively managing the situation.

4.5.3.1 Problems Encountered. Most of the problems faced in the development of the B-2 have been created by an inadequate understanding of the desired capabilities for the aircraft and the effort required to produce these capabilities. The major problems include:

Requirements Creep. The largest requirements creep occurred when the Air Force added the requirement for the B-2 to perform in a low level terrain following capacity as well as the initially planned high altitude role. This change required a major redesign of the airframe and the large amounts of software required to control it. In addition, new

CSCIs were created to handle the new terrain following capabilities. Requirements creep has been a way of life at the B-2 SPO, since the initial development of the requirements for the aircraft, many major changes have occurred. The changes have ranged from the mission change to the addition of the third crew members position to the incorporation of the latest technological breakthroughs. These changes will probably give the B-2 much greater functionality, but they have wreaked havoc on the software development process.

Immaturity of the Software Development Process. At the start of the program, Northrop basically had no standardized procedures for developing software or predicting either cost or schedule for a software development project.

Management Focus on Hardware. When the development of the B-2 began, management placed their focus on the development of the hardware that needed to be developed in order to accomplish some of the functions that were considered to be on the cutting edge of technology. They apparently felt the software development would be routine since the hard tasks would be accomplished in hardware. This misconception was corrected fairly early in process, but not early enough to avoid damage.

Underestimation of the magnitude of the Software Development Task. This problem was created partially ' e immaturity of Northrop's process, but also by the lack of software training received by both the personnel at Northrop and the SPO. The original estimates for both cost and schedule were surpassed long ago, but they were never really credible estimates in the first place. The B-2 management now has good grasp of the magnitude of the total task before them, but they are still having some trouble accurately estimating the effort required to perform some of the efforts required.

4.5.4 Case Study Summary. The majority of the problems faced by the B-2 program originated at the start of the program. The B-2 SPO and Northrop have done an excellent job of facing these problems and finding solutions. However, the program is still dealing with the aftermath caused by many of these problems, but the situation is getting

better instead of worse. The problem of requirements creep is still very prevalent in the program and no solution has been found. It will probably continue to be a problem as long as the program is in development.

The incremental design philosophy used in the development of the B-2 hardware and software has created both advantages and disadvantages. Air Force officials have decided that it is more important to build aircraft as soon as possible with a minimal capability than to have full functionality in the first production aircraft. However, this tradeoff comes with increased risk in certain areas such as configuration management. It also makes a program vulnerable to problems like requirements creep.

The B-2 program is the perfect example of how a program can start off on the wrong foot, but end up with successful results by placing proper attention on program improvement. Today, the program is not necessarily a model program, but still a very good one.

A snapshot picture of the problems earlier in the program is more clearly understood by the way the following cause and effect diagram fits them all together:

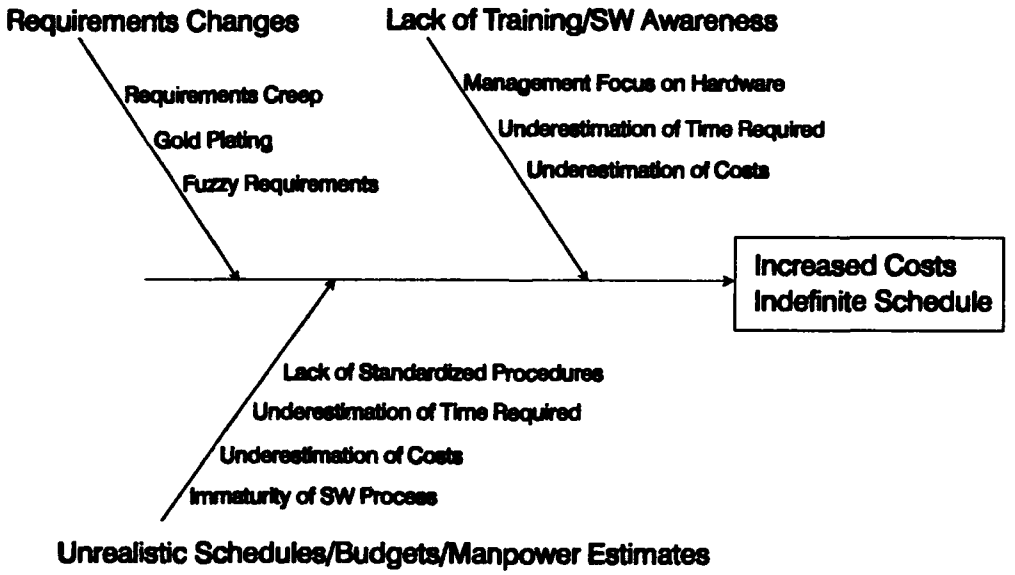


Figure 11. B-2 Cause and Effect Diagram

4.6 The F-22 Case Study.

4.6.1 Background. The F-22 is programmed to be the primary air superiority fighter for use by the United States Air Force into the next century. Basically, it is a replacement for the F-15 which is the current primary air superiority aircraft for the Air Force. The F-22 will integrate much of the state-of-the-art low-observable technology as well as a radically new avionics architecture.

4.6.1.1 Contract. The F-22 EMD contract was awarded to a team of contractors – Lockheed, Boeing and General Dynamics (now Lockheed, Fort Worth) - in August 1991. The contract was competed on the basis of a *fly-off* at the conclusion of the Demonstration/Validation phase between two teams who produced prototype aircraft. In keeping with the current Air Force philosophy of using cost-plus contracts when there is high risk/uncertainty, the contract was awarded on a cost-plus-award-fee arrangement. In this arrangement, the contractor earns base fee of 4% of the estimated cost of the contract, plus up to an additional 8% award fee based on overall contract performance (Morocco, 1991:44).

This award fee arrangement deserves closer examination. The award fee is determined semi-annually based on criteria established for that specific period. These criteria include cost, schedule, technical performance, and teamwork goals. Of specific interest is the inclusion of awards in at least one six month period based on adherence to the Software Development Plan and reusability of code produced. Clearly, software was a major consideration at the start of the contract based on this fact alone.

4.6.1.2 Program History. Demonstration/Validation of the prototype aircraft started in October 1986 with two contractor teams – one each for the F-22 and F-23. At the conclusion of Dem/Val the F-22 was chosen to be the aircraft for the next phase and in August 1991 the Engineering and Manufacturing Development contract was awarded to the joint venture team of Lockheed, Boeing and General Dynamics.

The F-22 program office, under Brigadier General Fain, was the pioneer in the using Integrated Process Teams (IPTs) to manage a program. These are joint contractor/government teams formed according to areas of responsibility which include members of all the necessary functional areas. IPTs are organized like miniature program offices with each team being led by co-leaders, one leader from the SPO and another from the contractor. One of the main focuses of this type of structure is to eliminate the adversarial relationship between the government and the contractor.

The F-22 program is seen by many to be a model program in the software area. Software was a primary consideration in the formation of both the Dem/Val and EMD contracts and its management seems to be an important considerations in the current management of the program.

4.6.1.3 Current Status. The program is currently between Preliminary Design Review (PDR) and Critical Design Review (CDR). PDR was conducted in April 1993 with CDR scheduled for November 1994. The program is currently undergoing a rephasing effort as a result of several factors. The Cold War drawdown resulted in the Air Force sacrificing two aircraft during EMD from eleven to nine, the Clinton Administration mandated new inflation rates be used in payment calculations, and a reallocation of funds in FY94 and FY96 due to budgeting constraints are forcing renegotiation of the contract.

4.6.2 Software Process.

4.6.2.1 Software Process Model. Significant effort was put forth at the outset of the EMD contract to define a software process to be used across all organizations developing software. This process is documented in the Software Development Plan (SDP) in accordance with Mil-Std 2167A.

The evolution of the SDP was well thought out by the program office as well as the prime contractor. At the outset of the EMD contract, the SPO conducted a capability/capacity review, in accordance with ASCP 800-5, of the prime contractor as

well as all of the subcontractors that would be contributing a significant amount of software on the contract. This review, which is similar to a Software Engineering Institute software process assessment, was used to choose a process model and software development plan which is consistent with the composite maturity of all the software organizations working on the contract. The basic process chosen was the waterfall model with a common methodology for real-time applications known as the Ada Design and Requirements Transformation System (ADARTS). Obviously, Ada was the language chosen for all applications where it was practical to use (approximately 95% of the code is Ada).

The importance of the process model and the SDP is evident in the way the SDP is used to manage software on the contract. Whereas many development contracts only require prime contractor adherence to the SDP, the F-22 contract has a very strict flow-down of SDP requirements to the subcontractors. Under this scheme the subcontractor SDPs must match paragraph for paragraph with the prime contractor SDP. Any subcontractor exceptions to the top level SDPs must be approved by the Government/prime contractor team known as the Computer Resources Control Board.

Much of the strict process definition mentioned above is necessary due to the PAVE PILLAR architecture. In this architecture, the majority of the processing for all of the aircraft subsystems is done in a centralized bank of computers known as the core computer. This core computer consists of a family of 14 standard processing and memory elements. Each subsystem is limited to using only these types of processors (Warner, 1993:3). Ease of maintenance is the primary reason for this architecture – there are fewer repair processes to learn with a reduced number of parts stocked both in the field and at the depot. Such an architecture requires a precise definition of interfaces as well as the capabilities of each processor type to ensure proper design of each of the subsystems.

4.6.2.2 Ensuring a Quality Software Product. Quality fielded system software was a primary concern from very early in the program. The Dem/Val contract made it very clear that any software carried over into the EMD phase would have to adhere to the same documentation and testing requirements as those in the EMD contract. Also, the program office stressed risk reduction during Dem/Val. This meant trying to demonstrate the ability to complete the harder tasks through prototyping. The result of these two factors is that little of the Dem/Val software was carried over into EMD. This means that the EMD software development will face a significant reduction in the risk of critical algorithms and the final EMD software will be of a higher quality and EMD.

Risk management is the key idea behind the F-22 SPO's trying to ensure the quality of the system software. The first step in this risk management is the identification of the risks. These risks are easily identified by the experience and lessons learned from other programs. Briefly, the problem areas the F-22 chose to address include:

Undisciplined Requirements Baselining Process
Inability to Project Realistic Schedules and Manpower
Questionable Capability, Capacity, Tools
Inadequate Development Process Discipline (Lyons, 1993:11).

The F-22 SPO has a plan for managing each one of these risk areas. To avoid an undisciplined requirements baselining process, the emphasis is placed on defining and controlling requirements more than the actual programming. A commitment to overall life-cycle planning helps to ensure that requirements are not forgotten or left out. The SPO sees requirements definition from everyone involved with the system – from developers and operators to supporters, testers and trainers – to be the first priority in the system development. Once defined, these requirements must be adequately documented in the program documentation – SORD, PDSSC, CRLCMP, and prime contract. However, even though requirements are in writing, the SPO understands that they are still subject to

change. This is managed by categorizing requirements using a volatility assessment to anticipate areas which are likely to change.

Projection of realistic schedules and manpower requirements are done by breaking down big jobs into manageable pieces. Complexity, type of application and contractor capability can then be paired with past experience to produce reasonable estimates. The Dem/Val phase served to validate this estimation methodology. Once these estimates are made they are continuously revised and monitored throughout the development using software management and quality indicators. These indicators are well defined and may change according to the current stage of the program.

Contractor capability, capacity and tools were evaluated partially in Dem/Val and at the beginning of EMD by the use of capability/capacity reviews. In addition, contractor self-assessment using the SEI methodology is encouraged so that upper management knows the capability of its own software organization.

The philosophy for maintaining development process discipline is to simply take no shortcuts. Hardware and software should be developed together with an integrated work breakdown structure and integrated master plans which define key system milestones. Thorough testing at each stage of development, strict configuration control, and adherence to the software development plan are key points to maintaining overall process discipline. The only way that these lofty goals can be met is by close interface with the contractor. The F-22 SPO does this by the use of Integrated Process Teams. This Government/contractor teaming arrangement ensures that everyone knows how the process works and management responsibility is assumed by the co-team leaders.

Probably the most effective way of ensuring software development process discipline is in the calculation of the six month award fee. The contractor's award fee is partially based on the program office's evaluation of how the contractor is conforming to the SDP.

This award fee arrangement is a significant incentive since upper management will surely be concerned with the bottom line.

4.6.2.3 Software Process Improvement Efforts. The capability/capacity reviews of the contractors by the SPO are a big first step toward improvement of the software processes in the individual software organizations. The problems must be understood before they can be corrected. Many of the contractors have performed self-assessments based on training from the SEI.

Once these problems or deficiencies are identified, an action plan can be identified and submitted for approval. Consistent with the Total Quality Management program a Process Action Team consisting of both Government and contractor personnel review the action plan. If approved, this may be written into the SDP or program documentation may be adjusted as necessary to implement the change. In addition, the SDP is subject to periodic review which may correct deficiencies.

4.6.2.4 Oversight by the SPO. The Integrated Process Team organization structure provides the SPO with a significant amount of information on the day-to-day development of the system. The risk management indicators alone give a detailed overview of the status of the software. This, in combination with the Government/contractor IPTs and the good working relationship with the contractor gives the SPO significant oversight and control of the software development.

4.6.3 Software Development Progress. There is little to report at this time on the progress of the software development. Since PDR just occurred in April 1993 and CDR is scheduled for November 1994, there are few hard products to evaluate. A detailed analysis of the SPO's software management indicators would be necessary to even identify potential problem areas. Although no specific problems are now evident, it is not uncommon for programs to have no significant software problems reported during this stage of development.

4.6.4 Case Study Summary. Although the F-22 has not experienced significant software problems to date, the SPO has identified specific risk areas based on previous experience and has implemented metrics to manage them. These risk areas neatly fit into the procurement practices and software development categories indicating that the program managers understand the importance of external factors affecting the development of the software and that they are familiar with the inner workings of the software development process. Until significant problems occur, no cause and effect diagram can be drawn. Although a cause and effect diagram could be drawn for the F-22 SPO's identified risk areas, such a diagram would not reflect any real problems the F-22 program is experiencing.

4.7 Survey Results

This section is organized according to the structure of the survey. First, each SPO will be evaluated independently using the results obtained in question 1 – significance of the 29 software problems listed in relation to each individual program. The problems identified from this question will be related back to the findings in the case study. Second, the results from question 2 will be grouped to produce an overall assessment of the software problems found in all major weapon systems. From this list of problems, the top ten will be selected for further evaluation. Third, an analysis of question 3 will determine the measurability of the top ten problems selected in question 2. Fourth, an analysis of question 4 will determine the correctability of the top ten problems selected in question 2. Fifth, from the analysis of questions numbers 3 & 4, the problems that will prove to be the best area for future process improvement will be determined.

4.7.1 C-17 Survey Analysis. The software personnel in the C-17 SPO chose not to complete any surveys in support of this thesis. This lack of cooperation is unfortunate

because the C-17 SPO would probably have received the greatest benefit from this effort based on the findings in the case study.

4.7.2 F-16 Survey Analysis – Question 1. The F-16 SPO has very few qualified software personnel. However, three surveys were received from the three qualified individuals. This section presents the significant information from the analysis of question 1 from these surveys. A complete statistical analysis of question 1 for the F-16 surveys can be found in Appendix C.

The experts from the F-16 SPO received a concordance rating of .4307 on question 1 using Kendall's coefficient of concordance. This value signifies an average value of concurrence and suggests that their agreement is strongly related, but not complete. This rating is not surprising since all 29 problems can not be expected to be major influences on every project. When problems are not significant, it is harder to determine which problem should be ranked higher than another. This situation results in a lower concordance rating. However, this rating should not preclude an examination of the top ten problems determined from this group. The top ten software problems in the F-16 SPO in order of significance are as follows:

1. Lack of training/SW awareness of contractor management
2. Unrealistic program schedules/budgets/manpower estimates
3. Immaturity of SW development process within the contractor organization
4. Requirements creep/gold plating
5. Lack of training/SW awareness of government management
6. Incorrect requirements/specifications
7. Changes in user requirements
8. Inadequate development facilities/tools
9. Unrealistic user requirements/performance goals
10. Underestimate of time required for SW testing and debugging

Table 2. F-16 Survey – Ten Most Significant Problems

These ten problems can all be related to three problem areas identified in the F-16 case study.

Requirements. Problems 4, 6, 7, and 9 all relate to the way the F-16 SPO identifies and processes user requirements. As mentioned in the case study, the F-16 SPO has built requirements creep into its acquisition process. These requirements are generated by a users group which may not be realistic in their decision making.

Immature Contractor Software Development Process. Problems 3, 8 and 10 relate to the lack of progress made by the contractor toward using updated software development processes and tools.

Lack of SPO Oversight and Inadequate Software Cost Tracking. Problems 1, 2, 5, and 10 relate to the lack of management attention toward software by both the contractor and the SPO. Without software cost tracking, no historical basis can be made for future software program estimates.

In the case of the F-16, the results of the survey associate very closely with the findings of the case study. This correlation validates the results of the F-16 case study.

4.7.3 B-2 Survey Analysis – Question 1. Kendall's coefficient of concordance for the B-2 problems was calculated to be .4549. This indicates a reasonable level of concordance amongst the 29 problems identified and was relatively high in relation to the question 1 concordance of the other two SPOs. Following are the top ten problems identified in question 1 for the B-2 SPO in order of significance:

1. Underestimate of time required for SW testing and debugging
2. Underestimate of time required for SW analysis/design/coding
3. Inadequate software development planning
4. Immaturity of SW development process within the contractor organization
5. Changes in user requirements
6. Low management priority toward SW early in development
7. Unrealistic program schedules/budgets/manpower estimates
8. Requirements creep/gold plating
9. Inadequate requirements analysis and review at major design reviews
10. Lack of training/SW awareness of contractor management

Table 3. B-2 Survey – Ten Most Significant Problems

As in the case of the F-16, the top ten problems relates closely to the problems identified in the case study.

Underestimation of the Magnitude of the Software Development Task. Problems 1, 2, 3, and 7 above all relate to the problem of underestimating the software development task identified in the case study. Software cost and schedule estimates were unrealistic from the beginning of the program.

Immaturity of the Software Development Process. Problem 4 was identified in the case study based on the lack of standardized procedures for software development and cost/schedule estimation. In fact, a lack of standardized software development techniques ranked eleventh on the list of problems which further supports the case study.

Requirements creep. Problems 5, 8, and 9 all represent the late change of the B-2 requirement to operate in a low-level terrain following capacity as well as the initially planned high altitude role.

Management Focus on Hardware. Problems 6 and 10 relate to the observation that upper management in the contractor organization placed little emphasis on software early in the development process. This may be due, in part, to a lack of software training/experience of upper management.

In the case of the B-2, the survey results match closely with the findings of the case study. The survey has therefore accomplished its purpose of validating these findings

4.7.4 F-22 Survey Analysis – Question 1. Although a top ten list of problems was identified for the F-22 SPO, the magnitude of these problems and their adverse impact on the program is relatively small. The Kendall's coefficient was the highest of the three SPOs at .46 indicating the highest level of agreement on the ranking of the problems. Most of the problems identified had a relatively small cost or schedule rating associated with them in relation to the B-2 and the F-16 problems. The only significant cost or schedule impacts identified are for the top two problems. Both of these problems could be the result of the rephasing effort which has caused the availability of fewer resources to the program. Most likely, this reduction in resources has not changed the schedule or cost expectations for the program by Congress.

1. Unrealistic program schedules/budgets/manpower estimates
2. Unrealistic user requirements/performance goals
3. Changes in user requirements
4. Requirements creep/gold plating
5. Excessive memory/throughput requirements
6. Shortfalls in hardware
7. Underestimate of time required for SW testing and debugging
8. Incorrect requirements/specifications
9. Underestimate of time required for SW analysis/design/coding
10. Lack of training/SW awareness of contractor management

Table 4. F-22 Survey – Ten Most Significant Problems

4.7.5 Composite Analysis – Question 2. The objective of this thesis is not only to analyze the problems from the individual programs, but to generalize these problems to all similar weapon systems. Question 2 gave the respondents the opportunity to voice what they think the major software problems are based on their overall experience, which may extend beyond the program on which they are currently working. The statistical analyses of the survey results of this question were quite interesting. The analysis of all 29 problems from all of the individuals surveyed showed a slightly negative concordance coefficient of -.0529. This would normally be interpreted as showing that there is no

agreement among the software professionals surveyed on the significant software problems. However, the analysis of the top ten problems shows a Kendall's coefficient of .4235. This is comparable to the concordance found in the individual programs which lends more credence to the results. Following are the top ten problems identified in order of significance:

1. Unrealistic program schedules/budgets/manpower estimates
2. Changes in user requirements
3. Underestimate of time required for SW testing and debugging
4. Low management priority toward SW early in development
5. Lack of training/SW awareness of government management
6. Inadequate requirements analysis and review at major design reviews
7. Inadequate software development planning
8. Immaturity of SW development process within the contractor organization
9. Incorrect requirements/specifications
10. Lack of training/SW awareness of contractor management

Table 5. Survey Composite Analysis – Ten Most Significant Problems

It should not be surprising that the top three problems in this list were also recognized in the top ten problems of question 1 from all three of the individual programs. This top ten list provides an area to focus attention. If a group of software professionals from three major weapon systems has agreed on the most significant 10 problems facing software development, these problems deserve closer examination. This will be accomplished in the following sections.

4.7.6 Problem Measurability – Question 3. When developing metrics it is important to be able to identify if the problem area can be accurately measured. If the problem can not be accurately measured it may be futile to collect metrics in that area. At the very least the manager should recognize that the data obtained from such metrics may be suspect. The cost of obtaining the data is also an important issue. It may be cost prohibitive to collect the data.

The measurability of the top ten problems from question 2 according to the software professionals surveyed is presented below. A rating of five or above indicates that the problem is quantifiable and that data can be made available with little additional effort. Ratings below this level indicate that the problem is subjective to analyze and that data is difficult or impossible to collect.

Problem	Measurability	Std Deviation
1. Unrealistic program schedules/budgets/manpower estimates	6.33	2.18
2. Changes in user requirements	7.11	2.37
3. Underestimate of time required for SW testing and debugging	5.89	0.93
4. Low management priority toward SW early in development	3.89	2.62
5. Lack of training/SW awareness of government management	5.33	1.73
6. Inadequate requirements analysis and review at major design reviews	4.67	0.71
7. Inadequate software development planning	5.11	2.20
8. Immaturity of SW development process within the contractor organization	4.67	2.29
9. Incorrect requirements/specifications	5.22	2.22
10. Lack of training/SW awareness of contractor management	5.67	2.12

Table 6. Measurability of the Ten Most Significant Problems

This data indicates that the top three problems are the most measurable since the average rating is above 5 and the standard deviations are small enough to indicate that the software professionals agreed that the problems are both quantifiable and that data is available with little additional effort. According to this criteria, these top three problem areas would be the best areas to develop metrics for.

4.7.7 Management's Ability to Change Problems – Question 4. Another important aspect of metrics is whether the problem can be corrected or not. If the problem can not be corrected, it may make little difference if metrics are used. The correctability of the top ten problems from question 2 according to the software professionals surveyed is presented below. A rating of 5 or above indicates that the problem can be corrected at low cost with little management attention. Ratings below 5 indicate either that the problem can only be corrected at a high cost with significant management attention or that the problem is uncorrectable.

Problem	Correctability	Std Deviation
1. Unrealistic program schedules/budgets/manpower estimates	6.44	2.13
2. Changes in user requirements	7.22	2.44
3. Underestimate of time required for SW testing and debugging	6.11	1.83
4. Low management priority toward SW early in development	7.89	1.17
5. Lack of training/SW awareness of government management	6.89	1.36
6. Inadequate requirements analysis and review at major design reviews	6.89	1.62
7. Inadequate software development planning	7.67	1.22
8. Immaturity of SW development process within the contractor organization	5.89	1.45
9. Incorrect requirements/specifications	5.33	2.06
10. Lack of training/SW awareness of contractor management	6.33	1.94

Table 7. Correctability of the Ten Most Significant Problems

These results are relatively good news. They indicate that the individuals surveyed think that each of the top ten problems can be corrected at a reasonable cost with only moderate management attention. In fact, this view was relatively consistent for most of the 29 problems identified in the survey. From this perspective, each of the top ten problems are good candidates for metrics to be established.

4.7.8 Overall Survey Analysis. As mentioned in chapter 3, the primary purpose of the survey was to validate the results of the case studies. This has been accomplished through the analysis of question 1. Question 2 provided a composite view of the significant software problems based on the overall experience of the software professionals surveyed.

The goal of this thesis is to identify the significant software problems on embedded systems so that metrics can be developed for the most important problems. This first step in the development of metrics is intended to be the limit of the scope of this thesis. However, the information collected from questions 3 and 4 of this survey concerning measurability and correctability of the problems may be useful in the further development of metrics for software development in subsequent research. Based on the preliminary research presented in this survey, problems 1, 2, and 3 are the best candidates for metric development. These problems have the desired measurability and correctability characteristics necessary for valid metrics.

Problem	Measurability	Correctability
1. Unrealistic program schedules/budgets/manpower estimates	**6.33**	**6.44**
2. Changes in user requirements	**7.11**	**7.22**
3. Underestimate of time required for SW testing and debugging	**5.89**	**6.11**
4. Low management priority toward SW early in development	3.89	7.89
5. Lack of training/SW awareness of government management	5.33	6.89
6. Inadequate requirements analysis and review at major design reviews	4.67	6.89
7. Inadequate software development planning	5.11	7.67
8. Immaturity of SW development process within the contractor organization	4.67	5.89
9. Incorrect requirements/specifications	5.22	5.33
10. Lack of training/SW awareness of contractor management	5.67	6.33

Table 8. Best Candidates for Metric Development – Ten Most Significant Problems

V. Conclusions and Recommendations

5.1 Restatement of Thesis

The question which this thesis attempts to answer is "What factors result in inadequate software, cost overruns, schedule delays, and other problems during the development of software in both Demonstration/Validation and Engineering and Manufacturing Development phases?" If these significant factors can be determined, a common set of metrics can be developed for use on all large weapon system developments in order to manage the risk in each of these areas.

In conducting the research to answer this question, the first step was to identify the possible categories in which these factors are rooted. Although software problems are often attributed to the software development process, external causes are also a possible source. As a result, the hypothesis to be validated by the research is that external factors such as government procurement practices and personnel deficiencies contribute to problems in software development just as much as deficiencies in the software development process.

5.2 Conclusions

A study which attempted to answer the same thesis question was requested by House Armed Services Committee in March 1992 and conducted by the GAO. Their report focused on fifteen specific C3I or embedded weapon systems in the DoD. This study took a similar approach, to this thesis, by dividing all software problems into three categories. However, it seems that this report did not take a very detailed look at the different programs, the list of problems is incomplete, and few software development

process issues were addressed. The following is a summary of the problems listed in the report under the three categories – management, requirements definition, and testing:

Management
- Lack of management attention/oversight
- Lack of adequate software management concepts, methods, practices
- Lack of adequate planning
- Development proceeded despite serious problems

Requirements Definition
- Lack of well-defined requirements
- Requirements change to meet new missions
- Lack of overall system perspective
- System not readily able to adapt to change
- Software products can not/may not meet security requirements

Testing
- Lack of adequate testing methods and approaches
- Lack of adequate system-level integration testing (GAO, 93-13:4)

5.2.1 Summary of Major Problems. This thesis also identified three different categories of problems. However, it focused on all software development problems instead of only the technical problems. The following is the list of software development problems identified by this thesis:

Acquisition Process
- Low management priority toward SW early in development
- Unrealistic program schedules/budgets/manpower estimates
- Requirements creep/gold plating
- Unrealistic user requirements/performance goals
- Changes in user requirements
- Incorrect requirements/specifications
- Undisciplined process to baseline requirements
- Lack of early user involvement
- Concurrent HW/SW development (HW schedule driving SW schedule)
- Shortfalls in hardware
- Management focus on hardware vs software

Software Development Process
- Inadequate requirements analysis and review at major design reviews
- Inadequate software development planning
- Underestimate of time required for SW analysis/design/coding
- Underestimate of time required for SW testing and debugging
- Lack of adequate metrics to measure SW development progress

Lack of standardized SW development techniques
Immaturity of SW development process within the contractor organization
Inappropriate process model used (waterfall, prototyping, spiral, etc.)
Inadequate development facilities/tools
Excessive memory/throughput requirements
Real-time performance shortfalls
Insufficient documentation

Personnel

Lack of training/experience of contractor SW programmers/engineers
Lack of training/SW awareness of contractor management
Lack of training/experience of government SW engineers
Lack of training/SW awareness of government management
Incompetence of contractor personnel
Incompetence of government personnel

Almost every potential problem identified in the procurement practices, software development process, and personnel categories was present to some extent on at least one of the programs studied. However, certain problems were found in nearly every case. These problems were found in three different areas. First, they were identified in the C-17, F-16 and B-2 case studies as significant problems. Second, the F-22 SPO identified most of these problem areas as high risk categories. Finally, the composite analysis of software problems by the survey identified these problems to be the most significant. In each category the most significant problem areas were as follows.

5.2.1.1 Procurement Practices.

Requirements. All aspects of the requirements process were recognized as problems for all of the programs. Requirements creep, unrealistic requirements, changes in user requirements, and incorrect requirements are all included in this area. The requirements process is fundamental to the acquisition process and has effects beyond the software arena when it does not function properly.

Management Focus on Hardware. Historically, management attention has been placed on the development of the hardware rather than the software, at least early in the program. This emphasis is starting to shift thanks to software failures of the past and the increasing

importance of software in the functionality of today's weapon systems. Also, the high cost of supporting software developed in the past which is unmaintainable and has poor documentation is now getting the attention of upper management within the Air Force.

Unrealistic Program Schedules, Budgets, And Manpower Estimates. Lack of experience with software on weapon systems and the increase in the number of functions provided by software both contribute to the problem of estimating the scope of a software development. The dynamic environment in which weapon systems are developed today also contributes to this problem. However, software estimation and measurement is emerging to become more of a science than an art. As estimation techniques are better defined and historical data is collected, improved budgets, schedules and manpower estimates can be made.

5.2.1.2 Software Development Process.

Inadequate Software Development Planning. Software development planning is a consistent problem for weapon system development from many different angles. In older programs, software development plans were either not required or not subject to Air Force approval. In more recent times, software development plans have either been inadequate or not followed by working level software personnel.

Immaturity of Software Development Process. Immaturity of the software development process is a problem which may be the root cause of many other problems. It is an area which has been attacked in different ways by different organizations. Aeronautical System Center uses a standardized procedure for evaluating the contractor's software capability/capacity for the purpose of source selection. The Software Engineering Institute conducts an assessment of the software organization to determine its maturity using their Capability Maturity Model. The purpose of this assessment is to define the level of maturity and facilitate process improvement. The fact that so much

effort is expended in the evaluation of contractor software development organizations is enough to verify that maturity of the process is a significant problem.

5.2.1.3 Personnel.

Lack of Training/Software Awareness of Contractor Management. Many of the individuals in contractor upper management have reached their position with no software background and no training in software management whatsoever. This creates a lack of awareness of the significant software issues in different stages of the weapon system development. Because of their lack of training and experience, these managers lack the tools to fix the problems once they are evident.

Lack of Training/Software Awareness of Government Management. The government currently has a critical shortage of experienced software professionals. Individuals are often chosen to manage software that have no experience or formal training. In addition, program directors may have no direct experience with software and most likely have never had any training in software management. Programs such as the Software Systems Management degree and the Software Engineering Professional Continuing Education courses at the Air Force Institute of Technology are attempting to fill the void of software professionals in the Air Force. Likewise, Project Bold Stroke was instituted to make Air Force upper management aware of software management issues through the use of a three day seminar.

5.2.2 Recommendations for SW Management. These are the problems areas which our study revealed as providing the most significant contributions to software cost overruns, schedule slips, and performance shortfalls amongst the four major weapon systems. The fact that significant resources are already being expended to alleviate these problems indicates an even greater need to measure progress to make sure that these resources are not being wasted. Now that the high risk areas are identified, standard risk management practices should be used to assess these areas on a regular basis on new

major weapon system developments. This means that metrics need to be developed which will indicate if these problems are an issue on a particular program as well as determining the effectiveness of risk mitigation efforts.

The SEI's Capability Maturity model does address these problems in Level 3 of their process improvement program. Therefore, an organization may find using the SEI's program the best route to improvement since it is an established program and will not require a trial and error approach.

5.3 Recommendations for further research

This thesis is only the first stage in the development of a metrics program which will allows the sufficient management of a major weapon system software development. From this point, metrics must be identified which will adequately measure the status of each of these problem areas. The most significant problem areas identified in this thesis should be the focus of the majority of the effort in developing these metrics. As mentioned in the survey results in Chapter Four, the measurability and correctability of the significant problem areas should also be taken into account when establishing a metrics program. The importance of this research is demonstrated by the House Armed Service Committees inquiries into software development problems and potential ways to improve development efficiency.

Another area for further research is to repeat the process performed by this thesis, but for systems other than major aircraft weapon systems. It is likely that the software development process varies over the types of weapon systems as the case studies have shown it to vary from flight control software to avionics software. The process found in this thesis must be repeated on these other classes of programs in order to facilitate real process improvement.

Appendix A: Acronyms

ADARTS.....Ada Design And Requirements Transformation System
AFSC.....Air Force Systems Command
ASC.....Aeronautical Systems Center
ASD .....Aeronautical Systems Division
ATARS.....Advanced Tactical Air Reconnaissance System
CE .....Concept Exploration/definition
CG.....Center of Gravity
CMM.....Capability Maturity Model
COTS .....Commercial Off-The-Shelf
CDR .....Critical Design Review
CPR.....Cost Performance Report
CPCI .....Computer Program Configuration Items
CPDS .....Computer Program Design Specifications
CRLCMP.....Computer Resources Life-Cycle Management Plan
CSCI .....Computer Software Configuration Item
DAC.....Douglas Aircraft Corporation
Dem/Val .....Demonstration and Validation
DoD .....Department of Defense
DSMC .....Defense Systems Management College
ECP.....Engineering Change Proposal
88th FTS.....88th Flying Training Squadron
EMD .....Engineering and Manufacturing Development
DT&E.....Developmental Test, and Evaluation
DPRO.....Defense Plant Representative Office
ENJJPT .....Euro NATO Joint Jet Pilot Training
FBWFCS .....Fly-By-Wire Flight Control Systems
FFP.....Firm Fixed Priced
FPIF .....Fixed-Price Incentive Fee
FMS .....Foreign Military Sales
FSD.....Full Scale Development
GAO.....General Accounting Office
GD .....General Dynamics
GE.....General Electric
LRIP.....Low Rate Initial Production
IPT .....Integrated Process Team
MMC.....Modular Mission Computer
MOU .....Memorandum Of Understanding
NATO .....North Atlantic Treaty Organization
PDR.....Preliminary Design Review
PDSSC .....Post Deployment Software Support Concepts

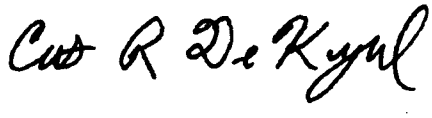
PMRT.....Program Management Responsibility Transfer
SDP.....Software Development Plan
SEI.....Software Engineering Institute
SEMP.....System Engineering Master Plan
SEPG.....Software Engineering Process Group
SEMS.....Systems Engineering Master Schedule
SIO.....Subcommittee on Investigations and Oversight
SORD.....System Operational Requirements Document
SPO.....System Program Office
SPR.....Software Problem Reports
TQM.....Total Quality Management
TRR.....Test Readiness Review

Appendix B: Survey

Survey to Analyze Root Causes of Problems
in the
Software Development Process
of
Major Weapon Systems

In an effort to apply Total Quality Management (TQM) to all aspects of the Air Force, momentum has developed to create metrics and measure everything possible. We believe that this not only creates a large quantity of meaningless data, but also obscures the data that is truly valuable. Our thesis intends to determine the root causes of software problems in the development of major weapon systems, and explore ways that these problems can be monitored in order to mitigate some of the risk in developing software for major weapon systems. Once the problems are identified, their use in a metrics program can be judged by whether they are both measurable and correctable. This survey is one part of our effort to discover these root causes. Thank you for your participation.

Capt Jay R. Hopkins
253-0121
jhopkins@afit.af.mil



1Lt Curtis R. De Keyrel
438-0480
cdekeyre@afit.af.mil

Personal Information

SPO_____ AREA_____

Years in SPO_____ Years of Software Experience_____

1. Rate the significance of the following software-related problems from 1 to 10 according to the adverse affect (10 = most adverse affect) each has had on your specific program (or the program you most recently worked) in relation to cost, schedule and performance of the system software according to the following scale. Rate the problem with the largest number from the three scales followed by the first letter of the scale (ie 8C, 7S, 9P). For example, if the cost rating is 6 and the schedule rating is 3, the response would be 6C. Please add any problems not listed.

Rating Scale

	1 2	3 4	5 6	7 8	9 10
Cost	Budget Estimates not Exceeded or Some $ Transfer	Cost Estimates Exceeded Budget by 1 to 5 percent	Cost Estimates increased by 5 to 20 percent	Cost Estimates increased by 20 to 50 percent	Cost Estimates increased in excess of 50 percent
Schedule	Negligible impact, change compensated by schedule slack	Minor Schedule Slip (less than 1 month)	Small Schedule Slip (1 to 3 months)	Development Schedule Slip in excess of 3 months	Large Slip that affects segment/system milestone
Performance	Minimal Performance Impact	Small Reduction in Technical Performance	Some Reduction in Technical Performance	Significant Degradation in Technical Performance	Technical Goals Can Not Be Met

	Rating
Acquisition Process	
Low management priority toward SW early in development	____
Unrealistic program schedules/budgets/manpower estimates	____
Requirements creep/gold plating	____
Unrealistic user requirements/performance goals	____
Changes in user requirements	____
Incorrect requirements/specifications	____
Undisciplined process to baseline requirements	____
Lack of early user involvement	____
Concurrent HW/SW development (HW schedule driving SW schedule)	____
Shortfalls in hardware	____
Management focus on hardware vs software	____
_____	____
_____	____
Software Development Process	
Inadequate requirements analysis and review at major design reviews	____
Inadequate software development planning	____
Underestimate of time required for SW analysis/design/coding	____
Underestimate of time required for SW testing and debugging	____
Lack of adequate metrics to measure SW development progress	____
Lack of standardized SW development techniques	____
Immaturity of SW development process within the contractor organization	____
Inappropriate process model used (waterfall, prototyping, spiral, etc)	____
Inadequate development facilities/tools	____
Excessive memory/throughput requirements	____
Real-time performance shortfalls	____
Insufficient documentation	____
_____	____
_____	____
Personnel	
Lack of training/experience of contractor SW programmers/engineers	____
Lack of training/SW awareness of contractor management	____
Lack of training/experience of government SW engineers	____
Lack of training/SW awareness of government management	____
Incompetence of contractor personnel	____
Incompetence of government personnel	____
_____	____
_____	____

2. Rate the significance of the following software-related problems according to the adverse affect (10 = most adverse affect) each has had in your overall experience in relation to cost, schedule and performance of the system software according to the following scale. Rate the problem with the largest number from the three scales followed by the first letter of the scale (ie 8C, 7S, 9P). For example, if the cost rating is 6 and the schedule rating is 3, the response would be 6C. Please add any problems not listed.

Rating Scale

	1 2	3 4	5 6	7 8	9 10
Cost	Budget Estimates not Exceeded or Some $ Transfer	Cost Estimates Exceeded Budget by 1 to 5 percent	Cost Estimates increased by 5 to 20 percent	Cost Estimates increased by 20 to 50 percent	Cost Estimates increased in excess of 50 percent
Schedule	Negligible impact, change compensated by schedule slack	Minor Schedule Slip (less than 1 month)	Small Schedule Slip (1 to 3 months)	Development Schedule Slip in excess of 3 months	Large Slip that effects segment/system milestone
Performance	Minimal Performance Impact	Small Reduction in Technical Performance	Some Reduction in Technical Performance	Significant Degradation in Technical Performance	Technical Goals Can Not Be Met

	Rating
Acquisition Process	
Low management priority toward SW early in development	____
Unrealistic program schedules/budgets/manpower estimates	____
Requirements creep/gold plating	____
Unrealistic user requirements/performance goals	____
Changes in user requirements	____
Incorrect requirements/specifications	____
Undisciplined process to baseline requirements	____
Lack of early user involvement	____
Concurrent HW/SW development (HW schedule driving SW schedule)	____
Shortfalls in hardware	____
Management focus on hardware vs software	____
_____	____
_____	____
Software Development Process	
Inadequate requirements analysis and review at major design reviews	____
Inadequate software development planning	____
Underestimate of time required for SW analysis/design/coding	____
Underestimate of time required for SW testing and debugging	____
Lack of adequate metrics to measure SW development progress	____
Lack of standardized SW development techniques	____
Immaturity of SW development process within the contractor organization	____
Inappropriate process model used (waterfall, prototyping, spiral, etc)	____
Inadequate development facilities/tools	____
Excessive memory/throughput requirements	____
Real-time performance shortfalls	____
Insufficient documentation	____
_____	____
_____	____
Personnel	
Lack of training/experience of contractor SW programmers/engineers	____
Lack of training/SW awareness of contractor management	____
Lack of training/experience of government SW engineers	____
Lack of training/SW awareness of government management	____
Incompetence of contractor personnel	____
Incompetence of government personnel	____
_____	____
_____	____

3. The purpose of this section is to determine the measurability of each problem area for use in a metrics program. Rate the measurability of each of the following problem areas from 1 to 10 according to the following scale.

Rating Scale									
1	2	3	4	5	6	7	8	9	10
Totally Subjective		Somewhat Subjective		Quantifiable		Quantifiable		Directly Quantifiable	
Data Collection Impossible		Data is very difficult to collect		Data Collection requires additional effort		Information can be derived from existing sources		Information readily available from existing sources	

	Rating
Acquisition Process	
Low management priority toward SW early in development	____
Unrealistic program schedules/budgets/manpower estimates	____
Requirements creep/gold plating	____
Unrealistic user requirements/performance goals	____
Changes in user requirements	____
Incorrect requirements/specifications	____
Undisciplined process to baseline requirements	____
Lack of early user involvement	____
Concurrent HW/SW development (HW schedule driving SW schedule)	____
Shortfalls in hardware	____
Management focus on hardware vs software	____
_____	____
_____	____
Software Development Process	
Inadequate requirements analysis and review at major design reviews	____
Inadequate software development planning	____
Underestimate of time required for SW analysis/design/coding	____
Underestimate of time required for SW testing and debugging	____
Lack of adequate metrics to measure SW development progress	____
Lack of standardized SW development techniques	____
Immaturity of SW development process within the contractor organization	____
Inappropriate process model used (waterfall, prototyping, spiral, etc)	____
Inadequate development facilities/tools	____
Excessive memory/throughput requirements	____
Real-time performance shortfalls	____
Insufficient documentation	____
_____	____
_____	____
Personnel	
Lack of training/experience of contractor SW programmers/engineers	____
Lack of training/SW awareness of contractor management	____
Lack of training/experience of government SW engineers	____
Lack of training/SW awareness of government management	____
Incompetence of contractor personnel	____
Incompetence of government personnel	____
_____	____
_____	____

4. Rate the ease with which each of these problem areas can be corrected with increased management attention or other actions from 1 to 10 according to the following scale.

Rating Scale

1 2	3 4	5 6	7 8	9 10
Uncorrectable regardless of mgt attention and cost	Correctable only with significant cost and management attention	Correctable at considerable cost	Correctable with increased mgt attention, minimal cost	Easily Corrected with minimal increase in mgt attention, no cost

	Rating
Acquisition Process	
Low management priority toward SW early in development	____
Unrealistic program schedules/budgets/manpower estimates	____
Requirements creep/gold plating	____
Unrealistic user requirements/performance goals	____
Changes in user requirements	____
Incorrect requirements/specifications	____
Undisciplined process to baseline requirements	____
Lack of early user involvement	____
Concurrent HW/SW development (HW schedule driving SW schedule)	____
Shortfalls in hardware	____
Management focus on hardware vs software	____
_____	____
_____	____
Software Development Process	
Inadequate requirements analysis and review at major design reviews	____
Inadequate software development planning	____
Underestimate of time required for SW analysis/design/coding	____
Underestimate of time required for SW testing and debugging	____
Lack of adequate metrics to measure SW development progress	____
Lack of standardized SW development techniques	____
Immaturity of SW development process within the contractor organization	____
Inappropriate process model used (waterfall, prototyping, spiral, etc)	____
Inadequate development facilities/tools	____
Excessive memory/throughput requirements	____
Real-time performance shortfalls	____
Insufficient documentation	____
_____	____
_____	____
Personnel	
Lack of training/experience of contractor SW programmers/engineers	____
Lack of training/SW awareness of contractor management	____
Lack of training/experience of government SW engineers	____
Lack of training/SW awareness of government management	____
Incompetence of contractor personnel	____
Incompetence of government personnel	____
_____	____
_____	____

Appendix C: Survey Question #1 (F-16 Specific Problems) Analysis

F-16 Question #1

			m= 3		
Acquisition Process	#1	#2	#3	Rj	Rj^2
Low management priority toward SW early in development	6	8.5	6	20.5	420.25
Unrealistic program schedules/budgets/manpower estimates	6	8.5	13	27.5	756.25
Requirements creep/gold plating	6	8.5	16	30.5	930.25
Unrealistic user requirements/performance goals	18	8.5	24	50.5	2550.25
Changes in user requirements	6	8.5	24	38.5	1482.25
Incorrect requirements/specifications	6	23.5	6	35.5	1260.25
Undisciplined requirements baselining process	6	19	14	39	1521
Lack of early user involvement	18	23.5	24	65.5	4290.25
Concurrent HW/SW development (HW schedule driving SW schedule)	18	8.5	24	50.5	2550.25
Shortfalls in hardware	18	8.5	6	32.5	1056.25
Management focus on hardware vs software	18	19	6	43	1849
Software Development Process					
Inadequate requirements analysis and review at major design reviews	23.5	14.5	24	62	3844
Inadequate software development planning	23.5	2	15	40.5	1640.25
Underestimate of time required for SW analysis/design/coding	23.5	1	17.5	42	1764
Underestimate of time required for SW testing and debugging	23.5	14.5	17.5	55.5	3080.25
Lack of adequate metrics to measure SW development progress	6	3	6	15	225
Lack of standardized SW development techniques	6	19	6	31	961
Immaturity of SW development process within the contractor organization	13.5	8.5	6	28	784
Inappropriate process model used (waterfall, prototyping, spiral, etc)	6	27	24	57	3249
Inadequate development facilities/tools	13.5	19	6	38.5	1482.25
Excessive memory/throughput requirements	28	8.5	6	42.5	1806.25
Real-time performance shortfalls	28	28.5	6	62.5	3906.25
Insufficient documentation	28	28.5	24	80.5	6480.25
Personnel					
Lack of training/experience of contractor SW programmers/engineers	13.5	19	24	56.5	3192.25
Lack of training/SW awareness of contractor management	6	8.5	12	26.5	702.25
Lack of training/experience of government SW engineers	13.5	19	24	56.5	3192.25
Lack of training/SW awareness of government management	6	19	6	31	961
Incompetence of contractor personnel	23.5	25.5	24	73	5329
Incompetence of government personnel	23.5	25.5	24	73	5329
					66594.5

$$W = 0.4307$$

(this equation described in section 3.4.4)

Appendix D: Survey Question #1 (B-2 Specific Problems) Analysis

B-2 Question #1

				m= 3	
Acquisition Process	#1	#2	#3	Rj	Rj^2
Low management priority toward SW early in development	15.5	8	6	29.5	870.25
Unrealistic program schedules/budgets/manpower estimates	15.5	8	6	29.5	870.25
Requirements creep/gold plating	15.5	2.5	18	36	1296
Unrealistic user requirements/performance goals	5.5	17	22	44.5	1980.25
Changes in user requirements	9.5	1	15.5	26	676
Incorrect requirements/specifications	15.5	25.5	20.5	61.5	3782.25
Undisciplined requirements baselining process	15.5	25.5	20.5	61.5	3782.25
Lack of early user involvement	21.5	17	28	66.5	4422.25
Concurrent HW/SW development (HW schedule driving SW schedule)	15.5	25.5	24.5	65.5	4290.25
Shortfalls in hardware	9.5	8	28	45.5	2070.25
Management focus on hardware vs software	5.5	25.5	13	44	1936
Software Development Process					
Inadequate requirements analysis and review at major design reviews	9.5	8	18	35.5	1260.25
Inadequate software development planning	3	8	6	17	289
Underestimate of time required for SW analysis/design/coding	1.5	8	6	15.5	240.25
Underestimate of time required for SW testing and debugging	5.5	2.5	6	14	196
Lack of adequate metrics to measure SW development progress	15.5	17	6	38.5	1482.25
Lack of standardized SW development techniques	5.5	17	15.5	38	1444
Immaturity of SW development process within the contractor organization	1.5	17	6	24.5	600.25
Inappropriate process model used (waterfall, prototyping, spiral, etc)	25	25.5	24.5	75	5625
Inadequate development facilities/tools	28	17	18	63	3969
Excessive memory/throughput requirements	28	8	24.5	60.5	3660.25
Real-time performance shortfalls	9.5	8	24.5	42	1764
Insufficient documentation	15.5	25.5	28	69	4761
Personnel					
Lack of training/experience of contractor SW programmers/engineers	21.5	17	6	44.5	1980.25
Lack of training/SW awareness of contractor management	21.5	8	6	35.5	1260.25
Lack of training/experience of government SW engineers	25	17	6	48	2304
Lack of training/SW awareness of government management	21.5	17	13	51.5	2652.25
Incompetence of contractor personnel	25	25.5	13	63.5	4032.25
Incompetence of government personnel	28	25.5	6	59.5	3540.25
					67036.5

W = 0.4549

(this equation described in section 3.4.4)

Appendix E: Survey Question #1 (F-22 Specific Problems) Analysis

F-22 Question #1

				m= 3	
Acquisition Process	#1	#2	#3	Rj	Rj^2
Low management priority toward SW early in development	4	22	23.5	49.5	2450.25
Unrealistic program schedules/budgets/manpower estimates	4	1	1	6	36
Requirements creep/gold plating	14.5	3.5	4	22	484
Unrealistic user requirements/performance goals	4	2	4	10	100
Changes in user requirements	14.5	3.5	2	20	400
Incorrect requirements/specifications	14.5	13.5	4	32	1024
Undisciplined requirements baselining process	14.5	22	7.5	44	1936
Lack of early user involvement	14.5	22	23.5	60	3600
Concurrent HW/SW development (HW schedule driving SW schedule)	14.5	16.5	16	47	2209
Shortfalls in hardware	4	13.5	16	33.5	1122.25
Management focus on hardware vs software	8.5	16.5	16	41	1681
Software Development Process					
Inadequate requirements analysis and review at major design reviews	14.5	22	7.5	44	1936
Inadequate software development planning	22	22	16	60	3600
Underestimate of time required for SW analysis/design/coding	8.5	10.5	16	35	1225
Underestimate of time required for SW testing and debugging	4	5.5	23.5	33	1089
Lack of adequate metrics to measure SW development progress	14.5	22	23.5	60	3600
Lack of standardized SW development techniques	22	22	23.5	67.5	4556.25
Immaturity of SW development process within the contractor organization	14.5	13.5	11	39	1521
Inappropriate process model used (waterfall, prototyping, spiral, etc)	27	8	23.5	58.5	3422.25
Inadequate development facilities/tools	22	8	11	41	1681
Excessive memory/throughput requirements	4	22	7.5	33.5	1122.25
Real-time performance shortfalls	4	28	7.5	39.5	1560.25
Insufficient documentation	27	22	23.5	72.5	5256.25
Personnel					
Lack of training/experience of contractor SW programmers/engineers	22	10.5	23.5	56	3136
Lack of training/SW awareness of contractor management	14.5	5.5	16	36	1296
Lack of training/experience of government SW engineers	27	12.5	16	55.5	3080.25
Lack of training/SW awareness of government management	22	8	11	41	1681
Incompetence of contractor personnel	27	28	23.5	78.5	6162.25
Incompetence of government personnel	27	28	23.5	78.5	6162.25
					67129.5

$$W = 0.4600$$

(this equation described in section 3.4.4)

Question #2

	F-16	F-16	F-16	F-22	F-22	F-22	B-2	B-2	B-2	m= 9 Rj	Rj^2	W= -0.052891808
Acquisition Process												
Low management priority toward SW early in development	24.5	16	1.5	2.5	7.5	23	19	11	6.5	111.5	12432.25	
Unrealistic program schedules/budgets/manpower estimates	12.5	16	1.5	2.5	2	1	11.5	11	6.5	64.5	4160.25	
Requirements creep/gold plating	28	16	15	24	7.5	4	11.5	3.5	13	122.5	15006.25	
Unrealistic user requirements/performance goals	24.5	16	28	11.5	7.5	4	19	20	23	153.5	23562.25	
Changes in user requirements	12.5	4	15	11.5	7.5	2	11.5	3.5	23	90.5	8190.25	
Incorrect requirements/specifications	12.5	1	15	2.5	15.5	4	19	27	19.5	116	13456	
Undisciplined requirements baselining process	2	11	21.5	24	7.5	6.5	19	20	19.5	131	17161	
Lack of early user involvement	12.5	16	15	24	23	23	11.5	11	19.5	155.5	24180.25	
Concurrent HW/SW development (HW schedule driving SW schedule)	2	16	28	24	18.5	15	5	27	16.5	152	23104	
Shortfalls in hardware	12.5	16	11	11.5	15.5	15	11.5	11	27	131	17161	
Management focus on hardware vs software	12.5	16	9.5	2.5	18.5	15	19	27	6.5	126.5	16002.25	
Software Development Process												
Inadequate requirements analysis and review at major design reviews	12.5	4	21.5	11.5	23	6.5	19	11	6.5	115.5	13340.25	
Inadequate software development planning	12.5	4	5.5	19.5	23	15	19	11	6.5	116	13456	
Underestimate of time required for SW analysis/design/coding	12.5	16	15	5	12.5	15	5	11	6.5	98.5	9702.25	
Underestimate of time required for SW testing and debugging	12.5	4	15	11.5	2	23	5	11	6.5	90.5	8190.25	
Lack of adequate metrics to measure SW development progress	24.5	8.5	5.5	11.5	23	23	5	20	6.5	127.5	16256.25	
Lack of standardized SW development techniques	12.5	8.5	21.5	19.5	23	23	19	20	6.5	153.5	23562.25	
Immaturity of SW development process within the contractor organization	24.5	4	21.5	11.5	15.5	10	11.5	20	6.5	125	15625	
Inappropriate process model used (waterfall, prototyping, spiral, etc)	12.5	16	15	28	7.5	23	19	27	19.5	167.5	28056.25	
Inadequate development facilities/tools	12.5	16	21.5	11.5	7.5	10	5	27	15	126	15876	
Excessive memory/throughput requirements	12.5	16	28	11.5	23	6.5	5	20	23	145.5	21170.25	
Real-time performance shortfalls	12.5	16	5.5	11.5	28	6.5	5	20	27	132	17424	
Insufficient documentation	29	16	5.5	11.5	23	23	1	20	27	156	24336	
Personnel												
Lack of training/experience of contractor SW programmers/engineers	12.5	16	21.5	19.5	12.5	23	25	20	27	177	31329	
Lack of training/SW awareness of contractor management	12.5	16	5.5	11.5	2	15	25	11	27	125.5	15750.25	
Lack of training/experience of government SW engineers	24.5	16	9.5	24	15.5	15	27.5	3.5	6.5	142	20164	
Lack of training/SW awareness of government management	12.5	16	5.5	19.5	7.5	10	25	3.5	6.5	106	11236	
Incompetence of contractor personnel	1	8.5	25.5	28	28	23	27.5	3.5	16.5	161.5	26082.25	
Incompetence of government personnel	24.5	8.5	25.5	28	28	23	29	3.5	14	184	33856	
											519828	

Question #2 Top 10

	F-16	F-16	F-16	F-22	F-22	F-22	B-2	B-2	B-2	m= 9 Rj	Rj^2	W= 0.423589024
Low management priority toward SW early in development	9.5	8.5	1.5	2	5	9.5	6.5	5.5	4	52	2704	
Unrealistic program schedules/budgets/manpower estimates	4.5	8.5	1.5	2	2	1	3	5.5	4	32	1024	
Changes in user requirements	4.5	4	15	6	5	2	3	1.5	9	50	2500	
Incorrect requirements/specifications	4.5	1	15	2	7.5	3	6.5	10	8	57.5	3306.25	
Inadequate requirements analysis and review at major design reviews	4.5	4	9.5	6	9.5	4	6.5	5.5	4	53.5	2862.25	
Inadequate software development planning	4.5	4	4	9.5	9.5	7.5	6.5	5.5	4	55	3025	
Underestimate of time required for SW testing and debugging	4.5	4	15	6	2	9.5	1	5.5	4	51.5	2652.25	
Immaturity of SW development process within the contractor organization	9.5	4	8.5	6	7.5	5.5	3	9	4	57	3249	
Lack of training/SW awareness of contractor management	4.5	8.5	4	6	2	7.5	9.5	5.5	10	57.5	3306.25	
Lack of training/SW awareness of government management	4.5	8.5	4	9.5	5	5.5	9.5	1.5	4	52	2704	
											27333	

Problem	F-22	F-22	F-22	F-16	F-16	F-16	B-2	B-2	B-2	Sum	Average	Std Deviation
Acquisition Process												
Low management priority toward SW early in development	2	1	3	5	3	3	5	3	10	35	3.89	2.62
Unrealistic program schedules/budgets/manpower estimates	8	8	8	5	8	6	5	2	9	57	6.33	2.18
Requirements creep/gold plating	8	8	7	7	8	8	5	7	4	62	6.89	1.45
Unrealistic user requirements/performance goals	6	1	7	7	5	4	6	2	3	41	4.56	2.19
Changes in user requirements	10	4	10	7	5	10	7	6	5	64	7.11	2.37
Incorrect requirements/specifications	10	3	5	7	6	3	4	4	5	47	5.22	2.22
Undisciplined requirements baselining process	4	2	7	7	3	5	6	4	3	41	4.56	1.81
Lack of early user involvement	4	1	3	3	2	5	6	7	3	34	3.78	1.92
Concurrent HW/SW development (HW schedule driving SW schedule)	5	2	5	5	9	4	6	5	4	45	5.00	1.87
Shortfalls in hardware	5	3	5	7	6	8	7	4	5	50	5.56	1.59
Management focus on hardware vs software	2	4	4	5	2	5	4	3	7	36	4.00	1.58
Software Development Process												
Inadequate requirements analysis and review at major design reviews	4	4	5	5	5	4	6	4	5	42	4.67	0.71
Inadequate software development planning	2	4	4	7	9	5	5	3	7	46	5.11	2.20
Underestimate of time required for SW analysis/design/coding	6	6	8	5	6	7	5	6	10	59	6.56	1.59
Underestimate of time required for SW testing and debugging	6	6	7	5	6	7	8	8	4	53	5.89	0.93
Lack of adequate metrics to measure SW development progress	8	3	5	5	7	7	6	6	7	54	6.00	1.50
Lack of standardized SW development techniques	2	6	5	5	8	4	5	6	7	48	5.33	1.73
Immaturity of SW development process within the contractor organization	2	3	8	5	7	5	6	5	1	42	4.67	2.29
Inappropriate process model used (waterfall, prototyping, spiral, etc)	2	3	4	5	5	5	6	4	3	37	4.11	1.27
Inadequate development facilities/tools	6	3	4	5	7	4	4	5	5	43	4.78	1.20
Excessive memory/throughput requirements	8	5	7	7	9	6	4	7	7	60	6.67	1.50
Real-time performance shortfalls	8	7	7	7	9	8	5	7	4	62	6.89	1.54
Insufficient documentation	4	6	5	7	8	8	5	7	5	55	6.11	1.45
Personnel												
Lack of training/experience of contractor SW programmers/engineers	6	4	5	9	9	5	4	9	5	56	6.22	2.17
Lack of training/SW awareness of contractor management	6	3	4	9	6	5	4	9	5	51	5.67	2.12
Lack of training/experience of government SW engineers	6	3	5	9	7	6	4	9	5	54	6.00	2.06
Lack of training/SW awareness of government management	6	3	4	9	6	6	4	5	5	48	5.33	1.73
Incompetence of contractor personnel	2	1	4	4	4	4	4	4	3	30	3.33	1.12
Incompetence of government personnel	2	1	4	4	4	4	4	4	3	30	3.33	1.12

Problem	F-22	F-22	F-22	F-16	F-16	F-16	B-2	B-2	B-2	Sum	Average	Std Deviation
Acquisition Process												
Low management priority toward SW early in development	8	7	9	8	6	6	6	10	7	71	7.89	1.17
Unrealistic program schedules/budgets/manpower estimates	6	5	9	3	9	8	4	7	7	58	6.44	2.13
Requirements creep/gold plating	9	7	9	8	7	9	5	2	7	63	7.00	2.29
Unrealistic user requirements/performance goals	9	7	9	8	7	8	8	7	7	70	7.78	0.83
Changes in user requirements	9	7	9	8	8	8	7	1	8	65	7.22	2.44
Incorrect requirements/specifications	7	3	7	8	7	3	4	6	3	48	5.33	2.06
Undisciplined requirements baselining process	8	6	8	7	7	4	7	7	8	62	6.89	1.27
Lack of early user involvement	7	4	9	9	9	2	6	10	7	63	7.00	2.65
Concurrent HW/SW development (HW schedule driving SW schedule)	6	5	6	7	5	4	10	5	5	53	5.89	1.76
Shortfalls in hardware	5	8	5	7	7	6	10	5	3	56	6.22	2.05
Management focus on hardware vs software	9	7	8	8	7	9	7	4	8	67	7.44	1.51
Software Development Process												
Inadequate requirements analysis and review at major design reviews	7	9	7	8	8	4	5	6	8	62	6.89	1.62
Inadequate software development planning	8	9	9	7	8	8	5	7	8	69	7.67	1.22
Underestimate of time required for SW analysis/design/coding	6	7	9	8	8	3	4	6	5	58	6.44	2.07
Underestimate of time required for SW testing and debugging	5	7	8	8	8	3	4	6	5	55	6.11	1.83
Lack of adequate metrics to measure SW development progress	8	7	9	8	8	3	4	4	5	56	6.22	2.22
Lack of standardized SW development techniques	7	7	7	7	8	3	4	3	5	51	5.67	1.94
Immaturity of SW development process within the contractor organization	8	7	5	7	7	4	4	5	8	59	5.89	1.45
Inappropriate process model used (waterfall, prototyping, spiral, etc)	7	5	6	7	4	3	5	5	3	45	5.00	1.50
Inadequate development facilities/tools	7	5	5	6	6	6	5	5	3	48	5.33	1.12
Excessive memory/throughput requirements	6	4	5	6	7	4	4	3	5	44	4.89	1.27
Real-time performance shortfalls	6	5	5	6	1	6	4	2	9	44	4.89	2.37
Insufficient documentation	8	5	7	7	6	4	6	6	9	58	6.44	1.51
Personnel												
Lack of training/experience of contractor SW programmers/engineers	6	5	6	7	7	7	4	7	3	52	5.78	1.48
Lack of training/SW awareness of contractor management	8	7	6	8	8	3	4	8	5	57	6.33	1.94
Lack of training/experience of government SW engineers	6	7	6	6	8	7	4	3	8	55	6.11	1.69
Lack of training/SW awareness of government management	8	7	6	6	8	8	4	7	8	62	6.89	1.36
Incompetence of contractor personnel	8	1	5	6	1	4	4	3	3	35	3.89	2.26
Incompetence of government personnel	8	1	5	6	2	2	4	3	8	39	4.33	2.60

Bibliography

"Achilles Heel. "*Aviation Week & Space Technology.*" 129: 15 (17 October 1988).

Aeronautical Systems Center (ASC). *Software Development Capability/Capacity Review.* ASCP 800-5. Wright-Patterson AFB OH: HQ ASC, 11 September 1992.

Aeronautical Systems Division (ASD). *Software Development Integrity Program.* Mil-Std-1803 (USAF). Wright-Patterson AFB OH: HQ ASC, 15 December 1988.

Air Force Systems Command (AFSC). *Software Risk Abatement.* AFSC/AFLCP 800-45. Andrews AFB MD: HQ AFSC, 30 September 1988

Air Force Systems Command (AFSC). *Software Independent Verification and Validation (IV&V).* AFSCP 800-5. Andrews AFB MD: HQ AFSC, 20 May 1988.

Air Force Systems Command (AFSC). *AFSC Software Quality Indicators: Management Quality Insight.* AFSCP 800-14. Andrews AFB MD: HQ AFSC, 20 January 1987

Buckley, Robert L. *An Analysis of Mission Critical Computer Software in Naval Aviation.* MS thesis, Naval Postgraduate School, Monterey CA. June 1991 (AD-A243 078).

Cannan, James W. "The Software Crisis," *Air Force Magazine,* 69: 46-52 (May 1986)

Charette, Robert N. *Software Engineering Risk Analysis and Management.* New York: McGraw-Hill Book Company, 1989.

Daniel, Wayne W. *Applied Nonparametric Statistics.* Boston: PWS-Kent Pub., 1990

de Fio, Pio and Anthony DeThomas. "Verification and Validation of Flight Critical Software." *AGARD, Software Engineering and Its Applications to Avionics.* Washington: AIAA, 1988.

"Defense Contract: Direct Hit." *Newsweek.* CII-15: 70 (14 April 1975).

Defense Systems Management College (DSMC). *Mission Critical Computer Resources Management Guide.* Washington: Government Printing Office, 1988

Department of Defense. *Defense Acquisition.* DoD Directive 5000.1. Washington: Government Printing Office, 23 February 1991.

Department of Defense. *Defense Acquisition Management Policies and Procedures*. DoD Instruction 5000.2. Washington: Government Printing Office, 23 February 1988.

Department of Defense. *Defense System Software Development*. DoD Standard 2167A. Washington: Government Printing Office, 29 February 1988.

Department of Defense. *Defense System Software Quality Program*. DoD-Std 2168, Washington: Government Printing Office, 29 April 1988.

Department of Defense. *Management of Computer Resources in Major Defense Systems*. DoD Directive 5000.29. Washington: Government Printing Office, 26 April 1976.

Department of Defense. *Technical Reviews and Audits for Systems, Equipments, and Computer Software*. DoD Mil-Std 1521B. Washington: Government Printing Office, 1 June 1976.

Department of Defense. *Use of Ada in Weapon Systems*. DoD Directive 3405.2. Washington: Government Printing Office, 30 March 1987.

Department of the Air Force. *Life Cycle Management of Computer Resources in Systems*. AFR 800-14. Washington: HQ USAF, 29 September 1986

88th Flying Training Squadron/DOTB (88th FTS). *Applied Aerodynamics*. F-V5N-B-AA-SW, P-V4A-N-AA-SW. Sheppard AFB TX: 88th Flying Training Squadron, July 1988.

Geiger, Clarence J, Michael H. Levy, and Albert E. Misenko. *ASD History: January - December 1977 Vol I*. Andrews AFB MD: AFSC History Office, 1978.

Geiger, Clarence J, Michael H. Levy, and Albert E. Misenko. *ASD History: January - December 1987 Vol I*. Andrews AFB MD: AFSC History Office, 1988.

Geiger, Clarence J. *Small Wonder: Development of the F-16 Fighting Falcon*. Andrews AFB MD: AFSC History Office, 1988.

General Accounting Office. *B-2 Bomber: Initial Flight Tests*, GAO/NSIAD-90-284. Washington: Government Printing Office, 1990.

General Accounting Office. *C-17 Transitioning to Production With Increased Schedule Risk*, GAO/NSIAD-89-195. Washington: Government Printing Office, 1989.

General Accounting Office. *Embedded Computer Systems: Significant Software Problems on C-17 Must be Addressed*, GAO/IMTEC-92-48. Washington: Government Printing Office, 1992.

General Accounting Office. *Embedded Computer Systems: Status of C-17 Software*, GAO/T-IMTEC-93-2. Washington: Government Printing Office, 1993.

General Accounting Office. *Information on the Complexity of C-17 Systems and Structures*, GAO/NSIAD-91-5. Washington: Government Printing Office, 1991.

General Accounting Office. *Mission Critical Systems: Defense Attempting to Address Major Software Challenges*, GAO/IMTEC-93-13. Washington: Government Printing Office, 1993.

General Accounting Office. *Status of the C-17 Program and Related Issues Affecting the McDonnell Douglas Corporation*, GAO/T-NSIAD-92-4. Washington: Government Printing Office, 1992.

General Accounting Office. *Military Airlift: Status of C-17 Aircraft Development Program*, GAO/NSIAD-92-205BR. Washington: Government Printing Office, 1992.

General Accounting Office. *Military Airlift: C-17 Supplier Management Problems Are Not Related to Budget Reductions*, GAO/NSIAD-92-207. Washington: Government Printing Office, 1992.

General Accounting Office. *Military Airlift: Status of the C-17 Development Program*, GAO/T-NSIAD-93-6. Washington: Government Printing Office, 1993.

Hall, Dana L. "Coping with the Crisis," *Aerospace Software Engineering*. 136: 607 Washington: American Institute of Aeronautics and Astronautics, 1991.

Humphrey, Watts S. *Managing the Software Process*. Reading MA: Addison-Wesley Publishing Company, 1989.

Kolcum, Edward H. "Fighter Effort Tests Collaboration." *Aviation Week & Space Technology*. 106: 47 (2 May 1977).

Larman, Brian T. "The Embedded Software Lifecycle - An Expanded View," *AIAA Computers in Aerospace Conference*. 2: 705-713. Washington: American Institute of Aeronautics and Astronautics, 1989.

Lyons, Robert P. Class Handout, Software Systems Management Colloquium. School of Systems and Logistics, Air Force Institute of Technology, Wright-Patterson AFB OH, 1 June 1993.

McCarty, Dyke and Edward Rowland, *The Acquisition of Major Systems*. School of Systems and Logistics, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, September 1991.

Merkley, Jeffery A. *The B-1B Bomber and Options for Enhancement*. Washington: Government Printing Office, 1988.

Mills, Harlan D. "Engineering Discipline for Software Procurement," *Proceedings of the IEEE 1987 Computer Assurance Conference*, 1: 1-5 (1987).

Morrocco, John D. "Air Force Awards Contract to Start Advanced Tactical Fighter Development," *Aviation Week and Space Technology*, 135: 44, (12 August 1991)

Morrocco, John D. "Cheney Proposes Stretchouts, Cuts in B-2, C-17 Programs," *Aviation Week and Space Technology*, 134: 44, (30 April 1990)

Reed, Fred. "STARS Director: Hardware Specs Stymie Software," *Federal Computer Week*, 37, (28 November 1988).

Ropelewski, Robert R. "Five-Nation Coproduction Under Way." *Aviation Week and Space Technology*, 106: 59, (2 May 1977)

Simmons, Ronald A. "Software Quality Assurance (SQA) Early in the Acquisition Process," *Proceedings of the IEEE 1990 National Aerospace and Electronics Conference NAECON 1990*. II: 664-669. New York: IEEE Press, 1990.

Sprent, Peter. *Applied Nonparametric Statistical Methods*. New York: Chapman and Hall, 1989.

Subcommittee on Investigations and Oversight (SIO). *Bugs in the Program - Problems in Federal Government Computer Software Development and Regulation*. Staff study, 101st Congress, 1st Session, 1989. Washington: Government Printing Office, 1989

"War Spurred Lightweight Fighter Effort." *Aviation Week & Space Technology*. 106: 71 (2 May 1977).

Warner, Ron, Darrell Wright, and Scott Vesper. An Assessment of the F-22 Program Software Development Process", Unpublished report. School of Systems and Logistics, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, 26 February 1993

Wheeler, Donald J. and David S. Chambers. *Understanding Process Control*. Knoxville TN: SPC Press, Inc., 1986

Wolf, Bruce R. *ASD History: January - December 1990 Vol I.* Wright-Patterson AFB OH: HQ ASD, 1990.

Yin, Robert K. *Case study research : design and methods.* Newbury Park CA: Sage Publications, 1991.

Vita

First Lieutenant Curtis R. De Keyrel was born on 22 April 1967 in Rochester, Minnesota. He graduated from Mayo High School in Rochester, Minnesota in 1985 and attended the University of Minnesota. He completed the Air Force Reserve Officer Training Corps program as a distinguished graduate and graduated with a Bachelors of Electrical Engineering in December of 1989. While waiting to be called to active duty, he worked as an associate computer programmer for Unisys in their Air Traffic Control Division. In November of 1989, he entered active duty at Sheppard AFB as a student pilot in the Euro NATO Joint Jet Pilot Training (ENJJPT) program. Upon graduation from flight training, he entered the Software Systems Management program at the School of Systems and Logistics, Air Force Institute of Technology, in May 1992.

Permanent Address:

1603 9th Avenue S.E.
Rochester, Minnesota 55904

Vita

Captain Jay R. Hopkins was born on February, 10, 1965 in Lewes, Delaware. He graduated from Cape Henlopen High School in Lewes, Delaware in 1983 and attended the U.S. Air Force Academy, graduating with a a Bachelor of Science in Management in 1988. Upon graduation, he received a regular commission in the USAF and served his first tour of duty at Wright-Patterson AFB, Ohio. There he served as a Project Manager for the Advanced Tactical Air Reconnaissance System (ATARS) in the Electronic Combat and Reconnaissance SPO where he was responsible for development of two of the major subsystems valued at $65 million. In his ATARS project management role he represented the Air Force in both military and national standards bodies for digital recording and was also tasked with interfacing with several joint and national programs. From there he was selected for the Software Systems Management curriculum at the School of Systems and Logistics, Air Force Institute of Technology, in May 1992.

Permanent Address:

RD 5 Box 213A
Lewes, Delaware 19958

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	December 1993	Master's Thesis

4. TITLE AND SUBTITLE
AN ANALYSIS OF THE ROOT CAUSES OF DELAYS AND DEFICIENCIES IN THE DEVELOPMENT OF EMBEDDED SOFTWARE FOR AIR FORCE WEAPON SYSTEMS

5. FUNDING NUMBERS

6. AUTHOR(S)
Jay R. Hopkins, Captain, USAF
Curtis R. De Keyrel, First Lieutenant, USAF

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Air Force Institute of Technology, WPAFB OH 45433-6583

8. PERFORMING ORGANIZATION REPORT NUMBER
AFIT/GSS/LSS/93D-2

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Major Tim Perkins
AFMC/ENSR
WPAFB, OH 45433

10. SPONSORING / MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION / AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*
The importance of embedded software, used in every subsystem of all major weapon systems used by the United States Air Force, has increased drastically over the last decades. However, in spite of the regulations currently in existence, developing and acquiring software which meets the user requirements within the original cost and schedule estimates continues to be difficult. At the same time, the Air Force has pushed to improve the development process with the Total Quality Management (TQM) program. The primary method used to improve the process has been to create metrics, collect data on these metrics, and then perform a statistical analysis on this data. This process has resulted in large quantities data, but very little improvement. This thesis executes the forgotten first step of process improvement – to analyze the process and determine the significant problems faced while developing software for embedded systems. This goal was accomplished by examining and evaluating four major acquisition programs: the B-2, C-17, F-16 and the F-22. In each of these programs, the problems identified are categorized as either procurement practice, software development process, or personnel issues. Of these, procurement practices caused at least as many problems as deficiencies in the software development process.

14. SUBJECT TERMS	15. NUMBER OF PAGES
Software Acquistion, Software Engineering, Software Problems, Software, Major Weapon System Development, B-2, C-17, F-16, F-22	125
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UL

AFIT Control Number AFIT/GSS/LSS/93D-2

AFIT RESEARCH ASSESSMENT

The purpose of this questionnaire is to determine the potential for current and future applications of AFIT thesis research. Please return completed questionnaires to: DEPARTMENT OF THE AIR FORCE, AIR FORCE INSTITUTE OF TECHNOLOGY/LAC, 2950 P STREET, WRIGHT PATTERSON AFB OH 45433-7765

1. Did this research contribute to a current research project?

   a. Yes b. No

2. Do you believe this research topic is significant enough that it would have been researched (or contracted) by your organization or another agency if AFIT had not researched it?

   a. Yes b. No

3. The benefits of AFIT research can often be expressed by the equivalent value that your agency received by virtue of AFIT performing the research. Please estimate what this research would have cost in terms of manpower and/or dollars if it had been accomplished under contract or if it had been done in-house.

   Man Years _____ $ _____

4. Often it is not possible to attach equivalent dollar values to research, although the results of the research may, in fact, be important. Whether or not you were able to establish an equivalent value for this research (3, above) what is your estimate of its significance?

   a. Highly Significant b. Significant c. Slightly Significant d. Of No Significance

5. Comments

_____ _____ _____

Name and Grade Organization

_____ _____

Position or Title Address

DEPARTMENT OF THE AIR FORCE
AFIT/LAC Bldg 641
2950 P St
45433-7765

OFFICIAL BUSINESS

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS MAIL PERMIT NO. 1006 DAYTON OH

POSTAGE WILL BE PAID BY U.S. ADDRESSEE

Wright-Patterson Air Force Base

AFIT/LAC Bldg 641
2950 P St
Wright-Patterson AFB OH 45433-9905