United States General Accounting Office

GAO

Report to the Committee on Armed Services, U.S. Senate

March 2004

# DEFENSE ACQUISITIONS

# Stronger Management Practices Are Needed to Improve DOD's Software-Intensive Weapon Acquisitions


GAO-04-393

March 2004

# DEFENSE ACQUISITIONS

# Stronger Management Practices Are Needed to Improve DOD's Software-Intensive Weapon Acquisitions

GAO Accountability • Integrity • Reliability

Highlights

Highlights of GAO-04-393, a report to the Committee on Armed Services, U.S. Senate

## Why GAO Did This Study

The Department of Defense (DOD) has been relying increasingly on computer software to introduce or enhance performance capabilities of major weapon systems. To ensure successful outcomes, software acquisition requires disciplined processes and practices. Without such discipline, weapon programs encounter difficulty in meeting cost and schedule targets. For example, in fiscal year 2003, DOD might have spent as much as $8 billion to rework software because of quality-related issues.

GAO was asked to identify the practices used by leading companies to acquire software and to analyze the causes of poor outcomes of selected DOD programs. GAO also was asked to evaluate DOD's efforts to develop programs for improving software acquisition processes and to assess how those efforts compare with leading companies' practices.

## What GAO Recommends

GAO recommends that the Secretary of Defense direct the military services and agencies to adopt specific controls to improve software acquisition outcomes. These practices should be incorporated into DOD policy, software process improvement plans, and development contracts. DOD concurred with two revised recommendations and partially concurred with two others.

www.gao.gov/cgi-bin/getrpt?GAO-04-393.

To view the full product, including the scope and methodology, click on the link above. For more information, contact Katherine V. Schinasi at (202) 512-4841 or schinasik@gao.gov.

## What GAO Found

Software developers and acquirers at firms that GAO visited use three fundamental management strategies to ensure the delivery of high-quality products on time and within budget: working in an evolutionary environment, following disciplined development processes, and collecting and analyzing meaningful metrics to measure progress. When these strategies are used together, leading firms are better equipped to improve their software development processes on a continuous basis. An evolutionary approach sets up a more manageable environment—one in which expectations are realistic and developers are permitted to make incremental improvements. The customer benefits because the initial product is available sooner and at a lower, more predictable cost. This avoids the pressure to incorporate all the desired capabilities into a single product right away. Within an evolutionary environment, there are four phases that are common to software development: setting requirements, establishing a stable design, writing code, and testing. At the end of each of these phases, developers must demonstrate that they have acquired the right knowledge before proceeding to the next development phase. To provide evidence that the right knowledge was captured, leading developers emphasize the use of meaningful metrics, which helps developers, managers, and acquirers to measure progress. These metrics focus on cost, schedule, the size of a project, performance requirements, testing, defects, and quality.

In a review of five DOD programs, GAO found that outcomes were mixed for software-intensive acquisitions. The F/A-18 C/D, a fighter and attack aircraft, and the Tactical Tomahawk missile had fewer additional cost and schedule delays. For these programs, developers used an evolutionary approach, disciplined processes, and meaningful metrics. In contrast, the following programs, which did not follow these management strategies, experienced schedule delays and cost growth: F/A-22, an air dominance aircraft; Space-Based Infrared System, a missile-detection satellite system; and Comanche, a multimission helicopter.

In response to congressional requirements, DOD, the military services, and the Missile Defense Agency have taken positive steps to improve the environment for acquiring software-intensive systems. However, their plans to implement software process improvement programs are not yet complete and more work is required to ensure controls that would help managers increase the chances of successful acquisition outcomes. Such controls include documenting baseline requirements agreements between the developer and acquirer that leverage systems engineering knowledge, meeting with the developer for periodic reviews (gates) during the development process, and obtaining meaningful metrics from the developer to manage the program. Furthermore, there are no assurances that program managers will be held accountable for using the plans once they are completed.

# Contents

## Figures



### Abbreviations

| | |
|---|---|
| CMM® | Capability Maturity Models |
| CMMI® | Capability Maturity Model Integration |
| CSC | Computer Sciences Corporation |
| DOD | Department of Defense |
| GSG | Global Software Group |
| MDA | Missile Defense Agency |
| NCR | National Cash Register Corporation |
| SA-CMM® | Software Acquisition Capability Maturity Model |
| SBIRS | Space-Based Infrared System |
| SW-CMM® | Capability Maturity Model for Software |

GAO
Accountability * Integrity * Reliability

**United States General Accounting Office**
**Washington, DC 20548**

March 1, 2004

The Honorable John W. Warner
Chairman
The Honorable Carl Levin
Ranking Minority Member
Committee on Armed Services
United States Senate

Computer software has increasingly become a critical component for Department of Defense (DOD) weapon systems. The development of complex software represents a potential leap forward in operational capability for any number of DOD defense acquisitions—from stabilizing a weapon to providing all of the key functions needed in an avionics system. Technological advancements have even made it possible for software to perform functions once handled by hardware. As the demand for complex software grows, the need for discipline while developing and delivering software also increases. In recent years, DOD has attributed significant cost and schedule overruns of software-intensive systems to difficulties in developing and delivering software. DOD estimates that it spends about 40 percent of its Research, Development, Test, and Evaluation budget on software—$21 billion for fiscal year 2003. Furthermore, DOD and industry experience indicates that about $8 billion (40 percent) of that amount may be spent on reworking software because of quality-related issues. We previously reported that DOD did not have effective and consistent corporate or software processes for software acquisitions, has had difficulty in implementing disciplined processes developed by industry experts, and some components had no software acquisition programs focused on improving processes and practices. We recommended that DOD correct these deficiencies by developing software process improvement programs.[1]

[1] See U.S. General Accounting Office, *DOD Information Technology: Software and Systems Process Improvement Programs Vary in Use of Best Practices*, GAO-01-116 (Washington, D.C.: Mar. 30, 2001). Recommendations contained in this report also called for specific components to consider basing their improvement programs on the Software Engineering Institute's IDEAL [SM] model. (IDEAL is a service mark of Carnegie Mellon University.)

In December 2002 Congress required the Secretaries of each military service and the head of those defense agencies that manage major defense software-intensive acquisition programs to develop process improvement programs for software acquisitions. Subsequently, the Senate Committee on Armed Services requested that we (1) identify the best practices and knowledge-based metrics used by leading companies to develop software, (2) analyze the causes of poor outcomes of selected DOD software-intensive acquisition programs, and (3) evaluate DOD's efforts to develop software process improvement programs and assess how those efforts compare with leading companies' practices to improve software acquisition processes.

## Results in Brief

The leading companies we visited focus attention on the software development environment, have disciplined development processes, and use metrics methodically to ensure that software is developed within cost, schedule and performance targets. Software acquirers, or organizations that purchase software, and developers work in an evolutionary environment where they are permitted to make incremental improvements to performance—rather than feeling pressured to set unrealistic expectations—and to strive to improve their software development processes on a continuous basis. This environment limits development to what is possible to manage. Software developers also are required to follow disciplined development processes. Each development phase—setting requirements, establishing a stable design, writing code, and testing—ends in a management review, or gate, to ensure that the project is on track. Additionally, software engineering requires peer reviews so knowledgeable staff can check each other's work and work together to remove defects at the earliest stage of development. To pass the management reviews, developers must demonstrate they have met the acquirer's expectations and quality standards before advancing to the next development phase. Having this knowledge in hand not only significantly increases the chances of successful outcomes but also helps leading companies identify opportunities to improve their software development processes over time. To track progress, confirm knowledge, manage risk, improve processes, and ensure that acquirers are well-informed, the leading developers we visited collect metrics from their development processes. These metrics include cost, schedule, size, requirements, tests, defects, and quality. By using these metrics, leading developers are able to maintain consistent development practices and quantify process outcomes.

In our review of five DOD weapon programs, we found that software outcomes were mixed. Software for the F/A-18 C/D, a fighter and attack aircraft, and the Tactical Tomahawk missile were very successful in meeting initial cost and schedule estimates. These programs emulated leading software development companies' practices. They were evolutionary, that is, they upgraded and fielded systems in incremental blocks of time, had achievable requirements that were managed carefully, required developers to gain the knowledge they needed to pass a management review at each gate in the development process, and collected meaningful metrics for management oversight. On the other hand, the following complex development programs experienced cost increases and schedule delays because of significant difficulties with software development: F/A-22, an air superiority and ground attack aircraft; Space-Based Infrared System (SBIRS), a missile-detection satellite system; and Comanche, a multi-mission helicopter. While each of these programs has been restructured with more oversight and has instituted more realistic controls over software requirements, each experienced significant requirements growth and cited that growth as a leading cause of development problems. Before restructuring the programs, neither the DOD program managers nor the software developers for these programs had a process with reliable reviews and deliverables to reduce development risk. None of the software developers for these programs were able to demonstrate sufficient use of metrics to track progress or whether the metrics they used were implemented consistently over time and used as a basis for comparison to provide oversight.

DOD's military services and the Missile Defense Agency (MDA) are at various stages of responding to the congressional direction[2] to improve software processes. The efforts so far provide a good starting point for changing the environment under which the services are managing software acquisition, but they are not complete. DOD's software acquisition practices could be strengthened by incorporating practices we found at leading companies, such as documenting agreements between the developer and acquirer that contain baseline requirements for the software developer based on systems engineering knowledge, meeting with the developer for gated reviews during the development process, and obtaining meaningful metrics from the developer to manage the program. Two other tasks assigned by Congress to DOD in the 2003 Authorization

[2] Bob Stump National Defense Authorization Act for Fiscal Year 2003, Pub. L. No. 107-314, sec. 804, Dec. 2, 2002. The text is in appendix III.

Act—setting criteria for how contractors are selected and establishing a best practices clearinghouse—are not yet complete.

We are making four recommendations to the Secretary of Defense to strengthen DOD's practices for managing software requirements, to ensure use of disciplined processes and metrics, and to include provisions in DOD's acquisition policy, plans, and contracts for improving outcomes of software acquisition. In written comments on a draft of this report, DOD concurred with two recommendations that we modified to incorporate wording that DOD suggested. The department partially concurred with two other recommendations. DOD agreed that the report provides useful insight for improving the software acquisition process and is consistent with the department's efforts to improve the process as it continues to implement section 804 of the Fiscal Year 2003 National Defense Authorization Act. DOD also agreed to take the report's findings into account as it monitors the process for continuous improvement and to apply our recommendations as further guidance to its component services and agencies.

## Background

DOD's major weapon systems rely more heavily on software to achieve their performance characteristics than ever before. According to information in a 2000 Defense Science Board Report, in the last 40 years, functionality provided by software for aircraft, for example, has increased from about 10 percent in the early 1960s for the F-4 to 80 percent for the F/A-22, which is currently under development.[3] The reasons for this are simple: performance requirements for weapon systems have become increasingly demanding, and breakthroughs in software capability have led to a greater reliance on software to provide more capability when hardware limitations are reached. Along with this, DOD's practice of expecting leaps in capability has placed extreme reliance on software development in most acquisitions. As DOD moves to more complex acquisitions—such as the integration of multiple systems in a single "system of systems"—understanding and addressing software development issues have become even more critical for DOD in order to control cost and deliver systems on time.

[3] Report of the Defense Science Board Task Force on Defense Software, Office of the Undersecretary of Defense, Acquisition and Technology, November 2000.

We have issued a series of reports on the knowledge that leading commercial firms gain and use to manage and control the acquisition and development costs of their products. Leading firms attain knowledge early in the development process about the technology they plan to incorporate and ensure that resources match requirements. They make sure the design is mature before approving production and have production processes under control before production begins. Implicit in this approach to product development is the successful development of software. Software is rapidly becoming a significant, if not the most significant, part of DOD's acquisitions. For example, software enables a missile to recognize a target; on some weapon systems, functionality as basic as flight is no longer possible without sophisticated software.

In addition to successful commercial practices and other significant resources that have proven effective for managing software acquisition and development, DOD has at its disposal numerous reports and recommendations by industry experts to transform DOD's software development process. This community of experts includes independent engineering teams, senior advisors on DOD's Defense Science Board, and Carnegie Mellon University's Software Engineering Institute. Although they have offered detailed guidance, DOD's software-intensive weapon system acquisitions remain plagued by cost overruns, schedule delays, and failure to meet performance goals.

## The Software Development Process

DOD is an acquisition organization—that is, it acquires major weapon systems and manages the overall acquisition process as well as the contractors who are tasked with developing the systems and associated software. The more managers know about software development processes and metrics, the better equipped they are to acquire software. On DOD's weapon system programs, the software development process is a part of the larger weapon system acquisition process. Software development has similar phases and—in the case of new systems—occurs in parallel with hardware development until software and hardware components are integrated. The following describes the four phases common to all software development:

**Determining requirements**: Software development begins with performance requirements for the component or for the fully integrated product. Ideally, a team of system and software engineers, users, acquirers or their representatives analyzes the overall requirements—operational characteristics, user interfaces, speed, maneuverability, survivability, and usability—and translates them into specific requirements, allocating some

to software and others to hardware. In more mature organizations, before making a commitment to develop a component or product, the software developer validates that the requirements allocated to software are realistic, valid, testable, and supportable. Management approves the requirements before the design phase begins.

Systems engineering, a comprehensive technical management tool, provides the knowledge necessary to translate the acquirer's requirements into specific capabilities. With systems engineering knowledge in hand, the acquirer and the developer can work together to close gaps between expectations and available resources—well before a program is started. Some gaps can be resolved by the developer's investments, while others can be closed by finding technical or design alternatives. Remaining gaps—capabilities the developer does not have or cannot get without increasing the price and timing of the product beyond what the acquirer will accept—must be resolved through trade-offs and negotiation. The basic steps in systems engineering include the following:

- defining what the acquirer wants, how the final product is to be used, what the operating environment will be, and what the performance characteristics are;
- turning the requirements into a set of specific functions that the system must perform; and
- identifying the technical and design solutions needed to meet the required functions.

Completion of these steps leads to a product design.

**Establishing a stable design:** The software development team develops a design that meets the software's desired functions. Numerous activities and documents typically are necessary to demonstrate that all of the software requirements are incorporated into a preliminary design and that functionality can be fully tested. The developer may construct a prototype for the acquirer to test the understanding of the requirements during the design phase. If management approves the preliminary design, the developer refines the design and managers conduct a critical design review before giving approval for the coding phase to begin.

**Manufacturing code:** Software code translates requirements and a detailed design into an executable series of instructions. In more mature software development organizations, developers are required to follow strict coding practices. These include ensuring that the code

- is reviewed by knowledgeable peers
- addresses requirements specified in the final design and
- follows strict configuration control procedures to ensure that no "secret code" is put in the system and generally follows coding documentation guidelines that enable software engineers other than the coder to understand and maintain the software.

**Testing to validate that software meets requirements:** To ensure that the design is ready for coding, testing activities start during the design phase and then continue through the coding phase. The testing of code is an important and critical phase and results in a series of quality-assurance tasks that seek to discover and remove defects that would hinder the software's performance. Completing these tasks requires the testers to coordinate with various stakeholders, such as the quality assurance group, to define test criteria that sufficiently test the approved software requirements.

## Resources for Quality Software Development

Significant resources are available to DOD for improving its software acquisition outcomes. Among these is Carnegie Mellon University's Software Engineering Institute, a federally funded research and development center. The Software Engineering Institute has identified specific processes and practices that have proven successful in fostering quality software development. The institute has constructed models for developing and acquiring software, developing and implementing software process improvement programs, and integrating hardware and software into a weapon system. To help organizations meet cost, schedule, and performance goals, the institute has issued guidance for adopting its models. The commercial firms we visited and DOD, both of which use the institute's models, consider them to be an industry standard. The institute created the models to provide general guidance for software development and acquisition activities that programs can tailor to meet their needs. These models can also be used to assess an organization's capability for developing or acquiring software.

The Software Capability Maturity Model[4]®, for example, focuses on improving software development processes. The model rates software maturity according to five levels of maturity:

[4] Capability Maturity Model is registered in the U.S. Patent and Trademark Office by Carnegie Mellon University.

- **Initial:** The software process is characterized as ad hoc. Success depends on individual effort.
- **Repeatable:** The basic process is in place to track cost, schedule, and functionality. Some aspects of the process can be applied to projects with similar applications.
- **Defined:** There is a standardized software process for the organization. All projects use some approved version of this process to develop and maintain software.
- **Managed:** The organization uses and collects detailed data to manage and evaluate progress and quality.
- **Optimizing:** Quantitative feedback about performance and innovative ideas and technologies contribute to continuous process improvement.

In addition, the institute has created a model specifically for software acquisition. This model follows the same five principles as the previous model but emphasizes acquisition issues and the needs of individuals and groups who are planning and managing software acquisition activities. A third model focuses on the integration of hardware and software and has a heavier emphasis in systems engineering. (See appendix II for a description of the three models.)

## Problems with DOD's Software Development Are Well Known

Despite acknowledgment of significant problems and access to extensive resources, DOD's problems with software acquisition have continued. In 2000 the Defense Science Board's Task Force on Defense Software reviewed selected DOD software-intensive systems and found that the programs lacked a well thought out, disciplined program management plan and software development process. The programs lacked meaningful cost, schedule, and requirements baselines, making it difficult to track progress. These findings are echoed by the work of DOD's Tri-Service Assessment Initiative, an independent group that evaluates Army, Air Force, and Department of Navy programs' software management processes and offers guidance for developing software in a disciplined manner. The Tri-Service Initiative found that three of the leading causes of problems in software-intensive systems are process capability, requirements management, and organizational management. A 1999 study performed by the Standish Group, an organization that researches risk, cost, and investment return for information technology investments, found that about one-third of software development programs—commercial or military—resulted in cancellation. Furthermore, in a series of studies completed through the 1990s, the group, found that the average cost overrun was 189 percent; the average schedule overrun was 222 percent of

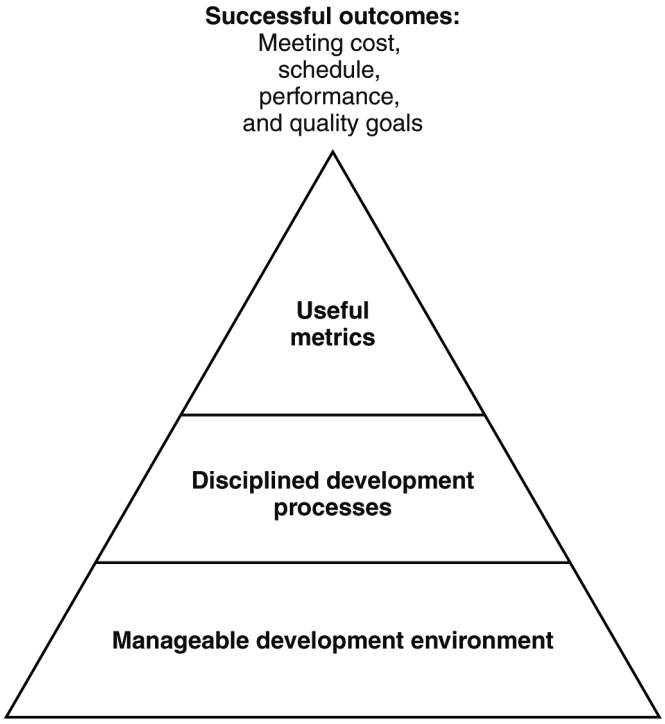
the original estimate; and, on average, only 61 percent of the projects were delivered with originally specified features or functions.

To address its problems with weapon acquisition, including software-intensive weapon systems, DOD recently revised its requirements generation and acquisition policies to incorporate a more evolutionary framework and improve its ability to deliver more capability to the acquirer faster.

## Successful Outcomes Are Largely the Result of Creating the Right Environment, Disciplined Processes, and Useful Metrics

Leading software companies we visited have been successful at software development largely because they establish a manageable product development environment, disciplined processes, and strong metrics to manage program outcomes. Key characteristics of a successful environment include evolutionary product development and continuous improvement of development capabilities so outcomes are more predictable. Within this environment, these companies use a structured management review process, and at the end of each of four key development phases—requirements, design, coding, and testing—the companies conduct reviews so that the development team does not progress to the next phase unless it attains a certain level of knowledge. A great deal of management attention is placed on the requirements-setting phase because missing, vague, or changing requirements tend to be a major cause of poor software development outcomes. Finally, leading developers we visited track cost and schedule outcomes with the help of a critical management tool, called earned value, a key indicator, or metric, for identifying and mitigating risk. In addition to earned value, developers use metrics for the size of a project, requirements, tests, defects, and quality to assess software development progress and to identify potential areas of improvement. Developers share this information with acquirers, who use the data to assess the risk software development has on overall product development and to make informed decisions about acquisitions. Figure 1 shows that a manageable environment, disciplined processes, and useful metrics are used together to form an effective process for software development.

**Figure 1: Key Management Practices That Increase Chances of Successful Outcomes**




Source: GAO's analysis of leading companies' software development practices.

## The Right Environment Reduces Software Development Risk

Three leading companies we visited—General Motors Powertrain Unit Motorola Global Software Group (GSG); and Teradata, a division of National Cash Register Corporation (NCR)—made a concerted effort to establish an environment that lowers risk and increases the chances of successful software development outcomes. This environment focuses on producing what is possible by establishing evolutionary product development while adhering to well-understood, well-defined, manageable requirements and encouraging continuous improvement of development processes. The environment enables leading companies to effectively compete in markets where delivery times are paramount and the acquirer expects reasonable prices and can go elsewhere with its business if not satisfied. Over time, these leading companies have learned that an evolutionary process emphasizing knowledge and quality enables successful outcomes. In comparison, an environment that allows too many risks, unknowns, and immature processes into product development can

have poor outcomes. In high-risk, low-technology maturity environments, developers find themselves forcing software to meet unrealistic expectations.

Officials at each of the companies we visited said that evolutionary product development is one of the fundamental elements of a manageable environment. Evolutionary development reduces risk because it allows software to be developed in small, manageable increments, with the availability of the complete software package coming later in the development life cycle. The General Motors Powertrain unit, which manufactures engines and transmissions, follows an evolutionary approach that calls for four to eight releases of the software product line each year. This approach offers many benefits, including allowing the software teams to restrict the size of projects to make them more manageable and to reduce risk. In addition, only well-defined requirements are included in the scope of the work, allowing the software teams to make improvements to previous releases.

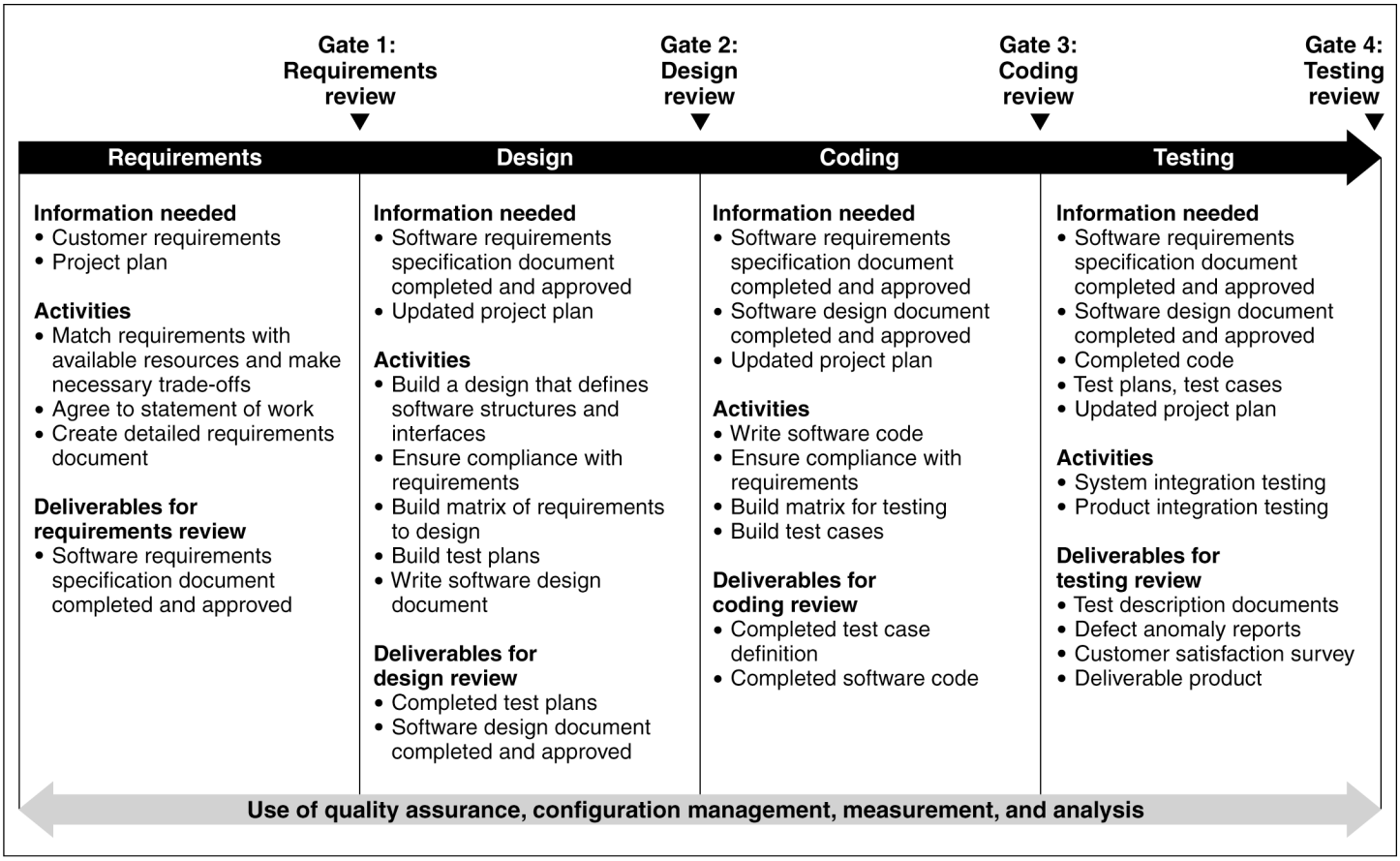
These leading companies consider continuous improvement to be an important part of their environment and culture, and most have implemented one of the Software Engineering Institute's Capability Maturity Models®. They have found that ad-hoc processes make it impossible to gain a clear understanding of when and how defects occur and make it difficult to fix processes so that the same defects can be avoided in the future. Motorola GSG officials told us it is not enough to hire talented software developers to achieve successful outcomes. Rather, companies must establish the right environment and use disciplined processes to help developers work efficiently and then target their recruiting efforts toward staff who can work in a process-oriented environment. This is not an easy task. Companies must be willing to invest time and money to develop new processes, collect meaningful data on a consistent basis, and train employees to follow the processes and interpret the data. In addition, management must display a strong commitment toward implementing the improved processes.

## Disciplined Software Development Processes Improve Software Outcomes

Within a low-risk, continuous improvement environment, leading companies we visited use a very structured, gated software development process that requires teams to obtain knowledge about the maturity of their software projects at key points in time. They plan, manage, and track activities for requirements, design, coding, and testing and rely heavily on such activities as configuration management, peer reviews, and quality assurance to help ensure the quality of their software. They also identify areas of risk and take actions to control the risks. Developers pay particular attention to the requirements-setting process because requirements are the foundation of a development effort. If requirements are not well defined or if there are too many changes, the result is additional, sometimes unmanageable risk.

Figure 2 is a general depiction of the process used by the companies we visited to manage software development. There are four development phases: determining requirements, establishing a stable design, manufacturing code, and testing to validate that the software meets the requirements and to detect errors. Within each phase are key activities that must take place and knowledge, or information, that must be attained to pass a review and move to the next phase of development.

**Figure 2: Highlights of the Knowledge-Based Software Development Process**




Source: GAO's analysis of leading companies' software development practices.

In addition to the four software development phases, these companies consider quality assurance, configuration management, measurement, and analysis to be integral parts of their software development activities. These activities assist developers in adequately managing software projects and collectively give the developer and the acquirer a level of confidence that the software is being developed within cost, schedule, performance, and quality targets. For example, configuration management allows developers to maintain a historical perspective of each software version change, keep a record of the comments made about the changes, and verify the resolution of defects. Quality assurance activities are typically focused on detecting and resolving defects. However, some companies, like Motorola GSG, may assign responsibility for detecting and

resolving defects to the project team and focus their quality assurance activities on evaluating whether project-associated work products adhere to the applicable process standards and procedures. In this case, quality assurance activities would also include ensuring that when the project teams do not comply with processes, these instances are identified, reported, and resolved at the appropriate level. Officials at each company we visited told us that the earlier defects are found and fixed, the less costly it is to the organization. If the defects are not found in the phase in which they occur, the cost to correct them grows in subsequent phases to the point where it could cost the company a significant amount of money to fix the problem once the software is fielded than if it had been corrected earlier.

## Requirements

Senior managers at software development and acquisition companies we visited expect requirements to be managed and controlled before design work begins and virtually all lower-level design elements to be adequately defined before the start of coding. Without adequate definition and validation of requirements and design, software engineers could be coding to an incorrect design, resulting in missing functionality or errors. Motorola GSG, a communications company, and Teradata, a division of NCR that specializes in database technology, estimate that about 95 percent of their requirements are set by the end of the requirements phase and 98 percent by the end of the design phase. Officials view managing requirements as the most critical development task to ensure successful software outcomes. They said that many software problems, often referred to as defects, could be traced to missing, vague, or changing requirements. Although company officials stated that some requirements-related defects are inevitable, such as those that arise when requirements are not sufficiently detailed, they said significant time and effort are necessary to elicit and document all requirements and determine the appropriate sequence for meeting these requirements. Nevertheless, mature organizations take time to conduct the various activities to sufficiently document and validate requirements before proceeding to preliminary design.

Leading software developers told us they typically devote about 20 to 30 percent of their software development time to requirements-setting activities. Doing so ensures that developers will be able to provide managers with key knowledge at the requirements review gate and show that requirements have been properly vetted with the acquirer and that they are achievable and well written. Activities they complete are highlighted below.

- **Establish integrated project teams:** Representatives from all acquirer and developer stakeholder groups use sound systems engineering techniques to establish software requirements.
- **Categorize requirements:** Acquirer and software team develop a comprehensive list of requirements and then categorize them on the basis of how critical they are to the product's performance.
- **Negotiate requirements:** Software team develops resource and schedule estimates on the basis of system engineering knowledge and past projects of similar size and scope. The software team then advises the acquirer which requirements may have to be delayed or sacrificed on the basis of resource and schedule goals.
- **Agree to requirements baseline:** Software team and acquirer agree to a requirements baseline that details the software requirements, including cost, schedule, performance, and quality goals the software team is expected to achieve.
- **Develop more detailed software requirements:** Using systems engineering, software team breaks the requirements into lower-level requirements, discusses the requirements with the acquirer, and formally documents the more detailed requirements.
- **Perform quality check:** Organization performs quality checks on requirements-related documents, such as the functional requirements document, to ensure that requirements are written clearly and all of the acquirer's requirements have been adequately addressed.

Company officials stress that to develop effective software requirements, the acquirer and developer must work closely together and have open and honest discussions about what can and cannot be done within desired time frames. Motorola GSG officials, for example, emphasize the importance of a written requirements baseline agreement with the acquirer to solidify software requirements and then strict adherence to requirements agreed to in order to avoid cost and schedule growth. They also perform detailed quality reviews to detect requirements problems early and to avoid costly rework in later stages.

Once developers establish requirements, they must also effectively manage the number and timing of requirements changes. Each developer we visited acknowledged that requirements could change at any point. However, officials told us that they aggressively manage requirements changes to make sure that they are reasonable and do not have a detrimental impact on project outcomes. For example, before making changes, they analyze the potential impact on cost, schedule, and performance and negotiate with the acquirer about whether the changes should be made within the ongoing project or in a future release. The

negotiation usually involves preparing an impact report for review by the acquirer or a governing board. Teradata, a division of NCR, goes further by limiting the number of changes it will make during the development cycle.

### Design

A stable design ensures that all requirements are addressed and that components and interfaces are defined. A Motorola GSG official stated that at least 90 percent of the company's software designs are stable before coding and suggested that developers that do not effectively manage the design phase could spend as much as 40 percent of a project's resources on rework activities. Leading companies complete a series of activities to stabilize their design and assure management that the software team is ready to advance to the next stage of development. These activities include, among other things, defining the overall functions and structure of the software on the basis of established requirements; selecting a system design; and developing the detailed system design specifications, which are sometimes referred to as the low-level design.

Typically, software teams will have two management reviews during this phase of development. A preliminary design review is used to examine the design rationale and design assumptions to ensure that the resulting software systems will meet the stated requirements. Particular attention is given to high-priority aspects of the system, such as performance, security, maintainability, and system recovery. User manuals and software test plans may also be examined at this time. A critical design review is conducted once the detailed design of the software system has been completed. The purpose of this review is to examine all design features to determine if they meet the acquirer's requirements. Throughout this phase companies typically perform peer reviews of design documents to detect errors and may also construct prototypes for the acquirers to test their understanding of the requirements.

### Coding and Testing

During the coding phase, software developers translate the requirements and design into a series of software steps that will control the system. According to company officials, well-written, achievable requirements, as well as very detailed designs, greatly enhance a software developer's ability to create software with relatively few defects. Additional processes that are critical to the success of this phase include peer reviews, coding standards, frequent unit testing, access to a library of pre-coded and tested functionality, and use of programming languages that enable the software engineer to document the code to facilitate understanding at a later time. For example, the leading companies we visited rely heavily on previously developed software to reduce development time, costs, and testing. According to company officials, it is not uncommon for them to reuse

70 percent of previously developed software on a new project. General Motors Powertrain officials emphasized that reuse is a top consideration for their projects and they have developed a software product line that teams use to complete requirements, design, and coding activities. Over the past few years, they have also re-engineered some of their electronic modules to allow for greater standardization of components within and across their Powertrain portfolio. This has greatly enhanced their ability to reuse software.

Testing is then performed to uncover defects or gaps in the code. Leading software companies we visited develop test plans after requirements are stable and take steps to ensure that there are one or more tests for each requirement. Through testing, teams assess the quality of the software to make it as defect-free as possible. For Motorola GSG, the software team is in control of all of the coding, testing, and quality-assurance activities. Officials stated that teams have access to online training and rely on libraries of previously used and tested code. They use peer reviews and inspections extensively during the requirements, design, and coding phases, for all software documents and test software and hardware components together to identify any integration problems that must be corrected.

## Metrics Provide Useful Insight to Software Development Activities

Leading developers we visited commonly use seven major types of metrics—cost, schedule, size, requirements, tests, defects and quality—to gauge a project's progress and identify areas for improvement. Acquirers use some of these same metrics to assess whether the developer will be able to deliver the software within cost, schedule, performance, and quality parameters.

We found that leading developers are relentless in their efforts to collect metrics to improve project outcomes and processes. The importance of metrics to these companies cannot be overemphasized. Motorola GSG and Teradata, a division of NCR, measure key aspects of software development for individual projects from the usual cost and schedule goals to process-improvement-type metrics that track the number and type of defects within each software development phase. They also have goals and metrics for companywide initiatives, such as cost-reduction efforts and customer satisfaction. Equally important, they have emphasized the critical nature of measuring processes, collecting metrics, and using them to analyze performance into their workforce through training.

Table 1 provides an overview of the seven categories of metrics used by the leading developers we visited, examples of their specific metrics, and how the companies use the metrics to manage their projects. Company officials cautioned that a variety of metrics could be used to satisfy each category listed in table 1 and that no one set of specific metrics would necessarily apply to all companies. Rather, companies tailor metrics from each category to fit their own needs.

**Table 1: Metrics Used by Leading Software Developers**

| Major metric | Examples of metrics used | Usefulness of metrics |
|---|---|---|
| Cost | • Cost and effort per phase<br>• Planned versus actual cost<br>• Cost performance index | Cost performance metrics, products of an earned value management system, indicate actual progress toward completing the software development against the plan. Large deviations between actual and estimated costs indicate that the project will have problems meeting cost and schedule goals. Management may have to consider taking actions such as reducing the scope of the project to meet release dates or even canceling the program. |
| Schedule | • Planned versus actual delivery dates<br>• Schedule estimation accuracy<br>• Percentage of project on time<br>• Schedule performance index | Schedule performance metrics, also products of an earned value management system, indicate achieved schedule progress against the plan. They are used throughout the software development phases to gauge progress toward developing key products or meeting critical milestones. Close attention to schedule deviations allows management to identify the team's ability to meet project goals and to determine if and when additional resources need to be added. |
| Size | • Amount of new, modified, and reused code<br>• Size estimation accuracy | Size metrics are used by management to compare the amount of code produced with the amount estimated. Changes to the size needed indicate potential cost and schedule problems. |
| Requirements | • Total requirements or features committed to deliver<br>• Percentage of requirements completed<br>• Number of requirements changes by phase | Requirements metrics are used to assess the organization's progress towards meeting the acquirer's performance demands. Developers try to avoid a large number of requirements changes or late changes because changes can impact cost and schedule commitments and can also result in software with a higher number of defects. |
| Tests | • Number of tests planned, completed, and passed<br>• Percent of planned tests completed | Test metrics are used to determine the extent to which planned software tests have been successfully accomplished. Deviations from the planned number of tests suggest that software might not have been adequately tested and may have quality problems, which could lead to costly rework in later phases. |
| Defects | • Number of defects per phase<br>• Phase defect originated versus phase found<br>• Cost to fix defect<br>• Severity of defects<br>• Total unresolved defects | Defect metrics are used to track problems with the software. Developers track defects to the phase where they were found, where they should have been found, and the cost to fix the problem. Large numbers of defects, particularly those that are found after the phase in which they were created, indicate performance problems that may lead to increased cost and schedule due to rework and the need to review development processes so that defects are found earlier. Identifying fewer defects than expected could also be problematic. For example, it may indicate that there is inadequate test coverage in the testing phase or that an insufficient formal technical review was performed on design documents in the design phase. |

| Major metric | Examples of metrics used | Usefulness of metrics |
|---|---|---|
| Quality | • Cost of quality efforts<br>• Cost of poor quality (rework)<br>• Number of quality goals missed and achieved<br>• Customer satisfaction survey results | Quality metrics provide information on the potential reliability of the delivered software and also provide an indication of the amount of money and time the developer invested in the development process in an attempt to assure a given level of quality. If defects are found and fixed during the phase in which they occur, this provides an indication that quality activities are performing well. If a defect is not identified in the phase in which it occurred, it becomes more expensive and time-consuming to fix and indicates weaknesses in the development process that need to be addressed. |

Source: GAO's analysis of leading companies' practices.

Leading developers we visited use metrics from each category above to actively oversee their projects and continuously assess their processes and projects to identify opportunities for improvement. Motorola GSG, for example, uses a standard set of metrics to enable project managers, as well as other levels of management, to assess the status of their individual software projects, staff productivity, requirements volatility, cost and schedule estimation accuracy, and the effectiveness of their quality assurance processes. Management also uses the information to compare similar projects within a software center or across the company to identify trends and areas that can be improved. They are particularly interested in tracking the number of defects by software development phase, the amount of rework associated with correcting the defect, and the amount of project resources spent to ensure quality. For example, data from one project show that developers were able to find and correct 92 percent of their problems during the phase in which they occurred. The other 8 percent were corrected by the end of the system test phase, resulting in only 1 percent of total project resources being spent to correct defects.

Motorola GSG uses an earned value management system to track the actual amount of time and effort it spends on project activities versus what it estimated for the projects. The earned value system, when properly implemented, provides developers and acquirers with early warnings of problems that could significantly affect the software project's cost and schedule. For example, according to private industry research, once a project is over 15 percent complete, developers will be unable to make up any overruns incurred to that point and the overruns will be even greater once the project is finished. This is often because project planning typically underestimates the time and effort required to implement planned tasks.

Motorola GSG uses a project time-tracking system to record the time spent on project activities attributed to the cost of quality and cost of poor

quality metrics. The cost of quality metric tracks the amount of time and money spent on such activities as formal quality reviews, testing, defect prevention, and rework to ensure a reliable product. If more resources were expended on these activities than expected, Motorola GSG would identify the reasons for this occurrence and improve its processes to try to prevent overruns from happening again. The cost of poor quality is also a concern to Motorola GSG because it quantifies the amount of rework that was necessary to address any product nonconformance, such as defects before (internal failure) and after (external failure) releasing the software product to the acquirer. According to company officials, the cost of poor quality is a direct reflection of the effectiveness of a company's software development processes. Generally speaking, poor processes lead to greater rework and a higher cost of poor quality, while better processes lead to a small amount of rework and a low cost of poor quality. Motorola GSG officials stated they have been able to hold the cost of poor quality (rework) to less than 5 percent for its projects by identifying when defects occur and then looking for improvements in their processes to try to prevent them from happening again.

Acquirers also need the types of metrics presented in table 1 to plan, manage, and track overall product development. These types of metrics allow acquirers to make their own assessments of the status of the software development project, where the software project is headed, the potential risk that software presents to overall product development, and if the developer's processes are effective in terms of reducing cost and schedule and improving quality. The earned value management system could provide acquirers with key information for calculating cost and schedule variations and also determining how much effort will be needed to complete a project on time when a project is behind schedule. If acquirers determine that software is likely to be late or over cost at completion, they then have the option to move some of the software requirements to a later development effort or allow the software development team more time to complete the project.

## Outcomes on DOD's Software-Intensive Acquisitions Were Influenced by Environment, Processes, and Metrics

In our reviews of five major DOD software-intensive weapon system acquisitions, we found mixed results. When DOD managers had a smaller, more evolutionary product with manageable requirements, used disciplined development process with gated reviews, and collected and used metrics to manage software development progress—such as the Tactical Tomahawk and the F/A-18-C/D programs—they delivered their product with less cost increase and less schedule delay. When DOD managers had expectations of developing revolutionary capabilities and did not use structured management reviews or collect and use metrics for software development—such as the F/A-22, SBIRS, and Comanche programs—they experienced significant cost growth and schedule delays. Table 2 illustrates how an evolutionary environment, effective process management, and use of meaningful metrics correlate with cost and schedule outcomes experienced by each program.

**Table 2: Program Outcomes Linked to Management Controls**

| Program | Evolutionary environment | Disciplined process | Use of meaningful metrics | Percent change in research, development, test, and evaluation cost estimate | Percent change in cycle time estimate |
|---|---|---|---|---|---|
| Tomahawk | Yes | Yes | Yes | 7.6 | 22.4 |
| F/A-18 C/D | Yes | Yes | Yes | 36.4 | 6.2 |
| F/A-22[a] | No | No | No | 127 | 104 |
| SBIRS[a] | No | No | No | 88 | Not available |
| Comanche[a] | No | No | No | 231 | 120 |

Source: GAO's analysis of DOD programs and selected acquisition reports.

[a]GAO's assessment of the evolutionary environment, disciplined process, and use of meaningful metrics addresses conditions found before these programs were restructured.

## Successful Outcomes for Two DOD Acquisitions

The Tactical Tomahawk and F/A-18 C/D programs were developed in an evolutionary environment, engaged in extensive work on requirements, controlled requirements' changes, collected and used detailed metrics to track development progress, and had less cost and schedule increase than the other programs we reviewed.

The Navy's Tactical Tomahawk missile will provide ships and submarines with enhanced capability to attack targets on land. New features include improved anti-jamming global positioning system, in-flight retargeting, and the ability to transmit battle damage imagery. Tomahawk program developers had disciplined development processes and used extensive peer reviews to discover defects and provided the acquirer with insight at each stage in development: requirements, design, code and test. They were

responsible for collecting and reporting data on a monthly basis, relying on metrics—cost, schedule, effort, size, requirements, testing, and defects that are similar to those used by leading commercial firms. The program office managed the acquisition based on the trends found in these metrics.

The F/A-18 C/D is a Navy attack fighter aircraft that has been deployed for a number of years. Periodically, the Navy upgrades the flight software to incorporate new features, add the capability to fire new munitions, and correct deficiencies discovered since the last upgrade. Working in an evolutionary environment, F/A-18 C/D program officials recognized that the success of the software upgrade to incorporate additional performance into the flight operations software depended on extensive requirements analysis before program start and firm control as requirements changed throughout development. This analysis ensured that the effort needed to meet requirements was well understood at the beginning of development, thus limiting the amount of redesign. Proposals for new requirements or changes to requirements after the program began were analyzed for cost, schedule, and performance impact. As with the Tomahawk program, FA-18 developers adhered to disciplined development processes, used extensive peer reviews to discover defects, and collected meaningful metrics to track progress.

## Outcomes Were Poor for Programs That Did Not Use an Evolutionary Approach, Disciplined Processes, and Meaningful Metrics

The F/A-22, SBIRS, and Comanche are complex programs that attempted to achieve quantum leaps in performance requiring extensive use of software rather than follow an evolutionary approach to software development. They all initially lacked controls over requirements, software processes, and metrics, causing major program upheavals. They encountered significant requirements changes, schedule slips, and cost increases because software defects were not discovered until later stages of the programs. Each of these programs has been restructured to incorporate requirements management controls, more-defined software development processes, and additional metrics.

The Air Force's F/A-22, originally planned to be an air dominance aircraft, will also have air-to-ground attack capability. It is expected to have advanced features, such as stealth characteristics, to make it less detectable to adversaries and capable of high speeds for long ranges. The F/A-22's avionics are designed to greatly improve pilots' awareness of the situation surrounding them. Early in the development process for the

F/A-22, we reported that the program's planned strategy for software development and acquisition was generally sound.[5] We cited the Air Force's plans to collect software costs and other software metrics to measure progress as examples of this sound strategy. At that time, we endorsed the program's plans to be event- rather than schedule-driven. However, as early as 1994, many features of this sound strategy were not being followed. Delayed software deliveries contributed to cost increases and schedule delays. Requirements and design changes accounted for 37 percent of the critical problem reports leading to avionics shutdowns in the F/A-22, according to program office reports. Program officials and contractor personnel agreed that requirements volatility had been a problem; however, they were unable to provide any specific measure of requirements changes because they had not tracked the overall growth in software requirements since the first 3 years of the program.

According to Lockheed Martin officials, the avionics system software is made up of 84 computer software configuration items,[6] each of which accounts for a specific avionics function, such as the interaction between the pilot and the aircraft. In our discussion with contractor and program personnel, they stated that disciplined processes in requirements control, design, testing, and configuration management were not uniformly followed because of cost and schedule pressures. The F/A-22 software strategy also called for the collection of software metrics to measure costs. Program and contractor officials were unable to provide metrics for sufficient management visibility over the overall progress of the software. The contractor stated that the Air Force did not compile metrics from lower levels into major segments such as avionics.

The Air Force's SBIRS satellites are being developed to replace DOD's older missile-warning satellites. In addition to missile warning and missile defense missions, the satellites will perform technical intelligence and battlespace characterization missions. Since the program was initiated in 1996, SBIRS has faced cost, scheduling, and technology problems. We have reported that SBIRS has experienced serious software design problems. Officials from Lockheed Martin, the prime contractor, stated that the program had uncontrolled requirements growth as well as overly

[5] U.S. General Accounting Office, *Air Force F-22 Embedded Computers*, GAO/AIMD-94-177R (Washington, D.C.: Sept. 23, 1994).

[6] A computer software configuration item is a software program that performs a common end-use function, follows its own development cycle, and is individually managed.

optimistic expectations about reusing software from a previous program. Program and contractor officials agreed that deficient systems engineering and the scarcity of personnel in software engineering disciplines contributed to ineffective control and to not understanding how much of the previous software could be reused. These officials also stated that neither the program office nor the contractor had a change management control process in place to analyze change requests. A thorough analysis late in the program revealed that very little of the software could be reused. Furthermore, because of a deficiency in resources devoted to systems engineering, the total requirements for the system were not adequately defined.

A report from an independent review team stated that more robust systems engineering could have precluded some of the problems. The report concluded that problems with the first SBIRS increment were primarily due to problems with software development and poor program execution. Peer reviews and engineering review boards were in place to monitor development, but, for reasons ranging from schedule pressures to reduced staffing, these decision bodies were ineffective. SBIRS contractor officials stated that they collected data on additions to requirements and on the number of lines of code, but because there were no restrictions on accepting new requirements and no control limits to the size of code, the metrics were not used to manage the project on a daily basis.

The Army's Comanche is a multi-mission helicopter intended to perform tactical armed reconnaissance. It is designed to operate in adverse weather across a wide spectrum of threat environments and provide improved speed, agility, reliability, maintainability, and low observability over existing helicopters. Since the program's first cost estimate, originally approved in 1985, the research and development cost for Comanche has almost quadrupled, and the time to obtain an initial capability has increased from 9 to over 21 years.

Several studies have identified software development as a problem area and highlighted requirements volatility and inadequate requirements analysis as having a large impact on the program. The lack of a disciplined process for Comanche's software acquisition was also cited as a reason for program shortfalls; however, the exact percentage of cost growth attributed to software is not known because the program office lacked adequate visibility into the software development process and, therefore, has little historical data on software. Comanche officials stated that initially they did not require a uniform set of metrics from the contractor.

They said they received earned value information from the contractor, but it combined software and hardware development data.

All three programs have been restructured and have instituted changes to bring more knowledge into the programs. For example, F/A-22 program officials report that their contractors have teamed with divisions within their companies that have more disciplined processes and they are reporting fewer problems with the avionics software. SBIRS program officials stated that they have instituted more controls over requirements changes, requiring analysis and approval at higher levels. Comanche officials reported that the program office has quarterly software reviews to focus attention on software development progress with the contractor and has adopted an incremental, block development strategy for software development. Program officials stated that they have asked for more-detailed metrics by which to manage the programs.

## DOD, the Services, and MDA Have Begun to Improve the Acquisition Environment, but Controls Needed to Assist Acquisition Managers

As a result of congressional requirements to initiate improvement plans and revisions to requirements and acquisition policies, DOD, the military services and MDA have created a more conducive environment for software acquisition and development. However, additional steps must be taken. We have found that leading software acquirers and developers we visited create disciplined software development processes and collect useful metrics for management oversight. These practices have proven to be a significant factor in their ability to achieve successful outcomes. DOD, the services, and MDA still lack controls in these areas that would put acquisition program managers in a better position to achieve successful program outcomes.

The plans that the services and MDA have begun in response to congressional direction have varying levels of detail and are at various stages of approval within the organizations. The Army, for example, has completed and has begun to implement its plan. The plan includes using pilot programs to provide information on metrics, and the Army expects to team with the Software Engineering Institute to identify training needs and continuous improvement. MDA has prepared a detailed draft that includes forming a baseline assessment of each missile defense element and making recommendations to the program office for each element to adopt improvement processes. MDA expects the elements to begin work once the baseline assessment is complete. The Navy's response includes teaming with the Software Engineering Institute to identify a course of action, including a training program for acquisition professionals and identifying software acquisition requirements and management initiatives.

The Air Force has called for a working group to begin in March 2004 to baseline Air Force practices and to suggest a course of action. These efforts establish an environment of change for the services and provide a platform upon which to make additional improvements. Furthermore, they make explicit to software an evolutionary approach to systems development and acquisition that DOD included in the recently revised requirements generation and acquisition policies.[7]

However, the services' and MDA's planning does not include practices we found at leading commercial firms that enable those firms to have successful outcomes. Furthermore, the plans do not incorporate controls that would ensure that the plans now being formulated are incorporated into acquisition practice. The plans could be strengthened by adding specific criteria to ensure that

- requirements' baselines based on systems engineering are documented and agreed to by both the acquirer and developer before a program's initiation and that cost/benefit analyses are required when new requirements are proposed;
- software developers and acquirers make efforts to continually improve practices over time;
- gated reviews and deliverables are integrated into the development processes; and
- developers collect and analyze metrics, including earned value to obtain knowledge about development progress and to manage risk.

Army, Navy, Air Force, and MDA officials said they have high-level support for improving software acquisition and for the plans they are developing, and the Army and MDA stated that they had included funding for software improvements in their budgets. Officials at the leading companies we visited emphasized that strong management support is needed to ensure success with process improvements. Although DOD has embraced an evolutionary approach in its acquisition policy, DOD has not yet incorporated a requirement specific to software process improvement into the policy. Furthermore, DOD has not said how it will require individual

[7] DOD Directive 5000.1, The Defense Acquisition System, describes the management principles for DOD's acquisition programs. DOD Instruction 5000.2, The Operation of the Defense Acquisition System, outlines a framework for managing acquisition programs. Collectively, these are known as the 5000 series. Chairman of the Joint Chiefs of Staff Instruction 3170.01C describes requirements generation policies and procedures of the Joint Capabilities Integration and Development System.

program offices to follow the guidance once the services and MDA establish full-fledged programs to improve software development processes.

Apart from the software acquisition improvement plans, DOD has taken some initiatives to strengthen software acquisition and development as well as address repeated performance shortfalls attributed to software. Since 1999 the Tri-Service Initiative has conducted detailed assessments of software-intensive programs to identity and mitigate software risks. The initiative has assessed about 50 programs spanning all military branches. While the results of individual initiatives are confidential to their programs, an overview shows three of the main causes of critical program performance problems: (1) the ability of the programs to establish and adhere to processes to meet program needs, (2) requirements management, and (3) organizational management. Process capability was a problem in 91 percent of case studies while problems with requirements management and organizational management were identified as problems 87 percent of the time. These findings are consistent with our discussions with leading companies about significant problem areas for software development management. This kind of information could prove useful to the military services and agencies as they plan for improving software acquisition. DOD has begun another initiative to strengthen the role that systems engineering plays in weapons system development as well as in software development. According to DOD officials, this initiative will include provisions for gated reviews of systems engineering baselines on an event-driven basis. Furthermore, the officials stated that they were working to incorporate the new systems engineering directives into acquisition policy.

DOD has tasked a source selection criteria working group with clarifying policy regarding source selection criteria for software-intensive systems, and another working group is creating a clearinghouse for best practices. The source selection criteria working group is discussing the application of software product maturity measures, and the Software Intensive Systems office is developing a proposal for a centralized clearinghouse of software best practices, but these initiatives are not complete.

To provide a better method of estimating the cost of software, DOD added a requirement to its acquisition policy to report such information as type of project, size, effort, schedule, and quality data to the Cost Analysis Improvement Group. DOD policy requires the Software Resource Data Report for major defense programs for any software development element with a projected software effort greater than $25 million.

## Conclusions

Organizations we visited that have established a strong, consistent, evolutionary environment and practices for setting product requirements, maintaining a disciplined development process, and using metrics to oversee development progress achieve favorable cost, schedule, and quality outcomes for software projects. These practices limit development efforts to what can be managed and result in decisions throughout the development process that are based on knowledge obtained through systems engineering that is sufficient to adequately gauge risks. The organizations we visited made business decisions to invest time and resources in achieving high process maturity levels to improve these practices. For the most part, in the programs reviewed, DOD garnered poor results from its software acquisition process because it has not employed consistent practices in these areas. Much as we have found in DOD's overall acquisition management process, the decisions to begin programs and to make significant investments throughout development are made without matching requirements to available resources and without demanding sufficient knowledge at key points. The acquisition programs we reviewed that used evolutionary environments, disciplined processes, and managed by metrics were more successful, and the programs that did not use these practices were less successful.

DOD has attempted to improve acquisition outcomes by establishing a framework for an evolutionary environment in its requirements generation and acquisition policies that develops manageable increments of capability. This is a positive step. However, DOD's policies do not contain the controls needed to ensure individual programs will adhere to disciplined requirements and development processes, nor do they include the metrics needed to do so. As DOD works to finalize its software process improvement plans, it has the opportunity to put in place those practices that have proven successful in achieving improved outcomes for software-intensive systems. In moving into a more complex, "system of systems" acquisition environment, much more will be demanded from software. The need for consistent practices and processes for managing software development and acquisition will become paramount if DOD is to deliver capabilities as promised.

## Recommendations for Executive Action

We have previously made recommendations to DOD to adopt certain specific practices developed by the Software Engineering Institute. As DOD changes the way it manages software intensive systems, it must take steps to ensure better acquisition outcomes. We recommend the Secretary of Defense take the following four actions:

- To assure DOD appropriately sets and manages requirements, we recommend that DOD document that software requirements are achievable based on knowledge obtained from systems engineering prior to beginning development and that DOD and the contractor have a mutual understanding of the software requirements. Furthermore, we recommend that trade-off analyses be performed, supported by systems engineering analysis, considering performance, cost, and schedule impacts of major changes to software requirements.
- To ensure DOD acquisitions are managed to a disciplined process, acquirers should develop a list of systems engineering deliverables (including software), tailored to the program characteristics, and based on the results of systems engineering activities that software developers are required to provide at the appropriate stages of the system development phases of requirements, design, fabrication/coding, integration, and testing.
- To ensure DOD has the knowledge it needs to oversee software-intensive acquisitions, we recommend that acquirers require software contractors to collect and report metrics related to cost, schedule, size, requirements, tests, defects, and quality to program offices on a monthly basis and before program milestones and that acquirers should ensure that contractors have an earned value management system that reports cost and schedule information at a level of work that provides information specific to software development.
- These practices should be included and enforced with controls and incentives in DOD's acquisitions policy, software acquisition improvement plans and development contracts.

## Agency Comments and Our Evaluation

DOD provided us with written comments on a draft of this report. The department concurred with two of the recommendations, subject to our incorporating some minor revisions. Since the suggested revisions did not materially change the intent of the recommendations, we revised them. For two other recommendations, the department partially concurred. The department agreed that the report provides useful insight for improving the software acquisition process and is consistent with its efforts to improve the process as it continues to implement section 804 of the Fiscal Year 2003 National Defense Authorization Act. It also agreed to take the report's findings into account as it monitors the process for continuous improvement and to apply our recommendations as further guidance to its component services and agencies.

The department further noted that the techniques highlighted in the report should not be seen as a panacea. We agree. Our report provides evidence that acquisitions can succeed if they take place in an evolutionary

environment rather than an environment that requires complex solutions for a single quantum leap in software capabilities. To augment an evolutionary environment, requirements must be carefully managed and existing systems and software engineering knowledge must be taken into account, the development processes must be disciplined and transparent to decision makers, and key metrics must be gathered and used to support decisions. We disagree with the department's observation that the report "plays down significant challenges associated with acquisition of complex defense systems …." To the contrary, our report highlights those challenges as inherent to acquisitions that proceed with limited knowledge about how to achieve quantum leaps in capability in a single acquisition. Our comparison of two successful evolutionary programs (Tactical Tomahawk and F/A-18 C/D, both categorized as major defense acquisition programs) with three revolutionary programs (F/A-22, SBIRS, and Comanche) shows different outcomes in terms of cost, schedule, and delivery of equipment to the warfighter.

DOD's rationale for providing programs with data less frequently than we recommended in our third recommendation suggested that data did not create knowledge and that knowledgeable software professionals are needed to interpret data. We agree that both knowledgeable people and data are needed, but those professionals must have data to interpret. We found that initially the F/A-22, SBIRS, and Comanche programs had knowledgeable staff but little data to analyze.

DOD indicated that it was already addressing software acquisition in policy in response to the fourth recommendation and cited multiple sections of DOD Directive 5000.1 as evidence. We do not agree that the current policy puts adequate controls in place to improve software practices to a level achieved by leading commercial companies. DOD is silent about including incentives in contracts for improving software processes. The department's comments are printed in appendix I.

## Scope and Methodology

To determine the best practices commercial companies use to manage software development and acquisition, we first conducted general literature searches. From these literature searches and discussions with experts, we identified numerous companies that follow structured and mature processes for software development and acquisition. We visited the following commercial companies:

**Computer Sciences Corporation (CSC)** develops individual business solutions for commercial and government markets worldwide. The

company is specialized in management and information technology consulting, systems consulting and integration, operations support, and information services outsourcing. In 2003, the company generated revenues of $11.3 billion. We visited CSC's Federal Sector office in Moorestown, New Jersey, and discussed its practices for developing and acquiring commercial and federal software. The Federal Sector unit has achieved a Level 5 Capability Maturity Model® rating.

**Diebold, Incorporated** manufactures self-service products, such as automated teller machines, electronic and physical security products, and software and integrated systems. In 2002 the company reported revenues of $1.9 billion. We visited the company's headquarters in North Canton, Ohio, and discussed the process it uses to develop software for automated teller systems.

**General Motors**, the world's largest vehicle manufacturer, designs, builds, and markets cars and trucks worldwide. In 2002 the company reported total net sales of $186.7 billion. We spoke with representatives from the Powertrain Group to discuss the processes used to develop and acquire electronic controls.

**Motorola GSG** provides integrated communications and embedded electronic solutions, such as wireless phones, two-way radio products, and internet-access products to consumers, network operators, commercial, government, and industrial customers. In 2002 the company reported net sales of $26.7 billion. We visited its Global Software Group offices in Montreal, Canada, and discussed the company's software and product development processes. The Global Software Group has achieved a Level 5 Capability Maturity Model® rating.

**NCR** offers solutions for data warehousing, retail store automation, and financial self-services. In 2002 the company reported sales totaling approximately $5.6 billion. We visited the Teradata Data Warehousing group office in San Diego, California, and discussed the software development process for the company's Teradata database software. The Teradata unit has achieved a Level 4 Capability Maturity Model® rating.

Software acquisition covers myriad activities and processes from planning and solicitation, to transition, to the support of a developed product. In fact, the Software Engineering Institute's Capability Maturity Models (CMM)® for software acquisition and development delineate more than a dozen different processes of this nature and offer principles governing the goals, activities, necessary resources and organizations, measurements,

and validation of each process. This report does not attempt to judge software acquisitions against all of those processes. Instead, our scope targets practices in three critical management areas we identified as problem areas from our previous work on weapon systems acquisitions and through discussions with leading companies. We limited our focus to ways to develop an environment that encourages continual improvement; improve the management of software development processes, including software requirements; and metrics to improve overall weapon system acquisition outcomes. In doing so, we borrowed criteria from each CMM® that offered a road map for continuous improvement in each of those specific areas.

At each of the five companies, we conducted structured interviews with representatives to gather uniform and consistent information about the practices, processes, and metrics that each company uses to manage software development and software acquisition. During meetings with representatives, we obtained a detailed description of the practices and processes they use to develop software within cost and schedule and ensure quality. We also consistently used a structured data collection instrument to collect metrics from the companies on their software projects. We met with company directors, software engineers, project managers, configuration managers, and quality assurance personnel.

Our report highlights several best practices in software development and acquisition on the basis of our fieldwork. As such, they are not intended to describe all practices or suggest that commercial companies are without flaws. Representatives from the commercial companies we visited told us that their practices have evolved over many years and that they continue to be improved on the basis of lessons learned and new ideas and information. This is not to say that the application and use of these practices have always been consistent or without error or that they subscribe to a single model for their practices and processes. However, they strongly suggested that the probability of success in developing and acquiring software is greatly enhanced by the use of these practices and processes.

We also selected five DOD weapon systems: RAH-66 Comanche, F/A-22, F/A-18 C/D, SBIRS, and Tactical Tomahawk. These systems are at various stages of development. We compared the practices, processes, and metrics the programs were using to manage software development and acquisition with the best practices commercial companies use. To identify the current policy, processes, and acquisition practices used in software development, for each program we visited, we conducted structured interviews with

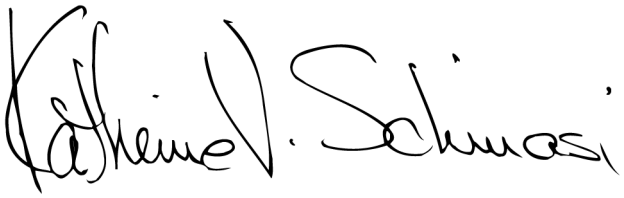
representatives from the program office and prime contractors Boeing Sikorsky for Comanche; Lockheed Martin, Marietta, Georgia, for F/A-22; and Lockheed Martin, Boulder, Colorado, for SBIRS. We also used a data collection instrument to determine which metrics program offices were collecting.

We selected Air Force, Army, and Navy programs because they all manage major defense acquisition programs. We also obtained the responses to date that the services and MDA have prepared in response to section 804 of the Bob Stump National Defense Authorization Act for Fiscal Year 2003. The legislation states that the Secretary of each military service and the head of each defense agency that manages a major defense acquisition program with a substantial software component shall establish a program to improve the software acquisition processes of that military service or defense agency. To determine how DOD responded to Congress's requirement, we met with DOD officials from the Tri-Service Assessment Initiative and the Software Intensive Systems Office and the staff responsible for developing the process improvement plans for the Air Force, Army, Department of the Navy, and MDA. We also met with officials from the Office of the Under Secretary of Defense (Acquisition, Technology and Logistics) concerning systems engineering initiatives and officials from the Office of the Assistant Secretary of Defense (Networks and Information Integration) concerning the software improvement plans. Because the plans are in varying stages of completeness, we did not evaluate to what degree the military services and MDA have complied with section 804. To determine whether the responses so far would help improve DOD's software acquisition, we evaluated the responses on the basis of the information we obtained from leading organizations concerning environment, disciplined processes, and collection of meaningful metrics.

We conducted our review between March 2003 and February 2004 in accordance with generally accepted government auditing standards.

We are sending copies of this report to the Secretary of Defense; the Secretaries of the Air Force, Army, and Navy; the Director of the Missile Defense Agency; and the Director of the Office of Management and Budget. We will also provide copies to others on request. In addition, the report will be available at no charge on the GAO Web site at http://www.gao.gov.

Please contact me at (202) 512-4841 if you have any questions concerning this report. Other key contributors to this report were Cheryl Andrew, Beverly Breen, Lily Chin, Ivy Hubler, Carol Mebane, Mike Sullivan, Sameena Nooruddin, Marie Penny Ahearn, Madhav Panwar, and Randy Zounes.



Katherine V. Schinasi
Director, Acquisition and Sourcing Management

# Appendix I: Comments from the Department of Defense

ACQUISITION, TECHNOLOGY AND LOGISTICS

OFFICE OF THE UNDER SECRETARY OF DEFENSE

3000 DEFENSE PENTAGON
WASHINGTON, DC 20301-3000

FEB 20 2004

Ms. Katherine V. Schinasi
Director, Acquisition and Sourcing Management
U.S. General Accounting Office
Washington, D.C. 20548

Dear Ms. Schinasi,

The Department of Defense (DoD) offers the following response to the GAO draft report, "WEAPONS ACQUISITIONS: Strong Management, Processes, and Metrics Needed to Improve Software Acquisition," dated January 29, 2004, (GAO Code 120215/GAO-04-393).

The draft report provides useful feedback and insight into software acquisition areas that require improvement, and in fact is consistent with the Department's current efforts to re-vitalize the state of the practice across the defense acquisition community. We will take the findings from this report into consideration as we continue to monitor our software acquisition process improvement activities, and continuously improve our acquisition practices. We further agree with findings depicting the Department's implementation of the FY03 National Defense Authorization Act, Section 804 as progressing in accordance with legislated requirements. Specific recommendations relating to Section 804 will be applied as further guidance to improvement programs within the Department's Component Services and Agencies.

Although we concur with comments and amplifications to the report's recommendations, it is important to note that while the specific techniques highlighted may improve insight and impose better controls over software development, they should not be seen as a panacea. In fact, the report plays down significant challenges associated with acquisition of complex defense systems by generalizing software development across programs. It is important to take into account a program's acquisition characteristics (size, complexity, stage of development, technology precedent, team makeup, length of service) before concluding, as this report does, that the difficulties experienced by DoD's complex developmental programs (e.g. F22, Comanche) were primarily due to failure to implement software best practices of commercial wireless phone manufacturers and database-software developers.

Detailed comments to specific recommendations made in this report are provided at TAB A. Under separate cover, additional technical comments have been forwarded to your staff to correct/clarify information in selected sections of the report.

Glenn F. Lamartin
Director
Defense Systems

Enclosure:
As stated

**GAO DRAFT REPORT - DATED JANUARY 29, 2004**
**GAO CODE 120215/GAO-04-393**

**"WEAPONS ACQUISITIONS: STRONG MANAGEMENT, PROCESSES, AND METRICS NEEDED TO IMPROVE SOFTWARE ACQUISITION"**

**DEPARTMENT OF DEFENSE COMMENTS TO THE RECOMMENDATIONS**

**RECOMMENDATION 1**: We recommend that DOD document that software requirements are achievable based on knowledge obtained from systems engineering prior to beginning development and that DOD and the contractor have a mutual understanding of the software requirements. Further, we recommend that costs and benefits of major changes to software requirements are supported by systems engineering analysis. (revised recommendation, ref. email from GAO received 4 Feb 04)

**DOD RESPONSE:** Partial concur. We concur with the recommendation with modification of the last sentence to read "Further, we recommend that *tradeoff analyses be performed, supported by systems engineering analysis, considering performance, cost, and schedule impacts* of major changes to software requirements." This acknowledges that performance, schedule and cost impacts, are all considered when evaluating requirements changes.

The report supports this recommendation with description of the commercial sector's ability to freeze 95% of their requirements by the end of the requirements phase, 98% by the end of design. The DoD must retain some flexibility for rapid development, prototyping and experimentation to realize objectives of force transformation. Military software functionality and associated requirements are sometimes subject to unpredictable and unforeseen perturbations due to changes in global capabilities and the attendant need for systems to adapt to such changes and technological enhancements.

**RECOMMENDATION 2:** The GAO recommended that the Secretary of Defense ensure that DoD acquisitions are managed to a disciplined process by developing a list of software knowledge deliverables based on the completion of systems engineering that contractors are required to provide at the end of software phases, requirements, design, coding and testing. (p. 32/GAO Draft Report)

**DOD RESPONSE:** Partial concur. We would modify the second recommendation with the italicized wording to read "... the Secretary of Defense ensure DoD acquisitions are managed to a disciplined process by developing a list of *systems engineering* deliverables *(including software), tailored to the program characteristics*, and based on the *results* of systems engineering *activities* that *software developers* are required to provide at the *appropriate stages* of the *system development* phases of requirements, design, *fabrication/coding*, *integration* and testing".

Deliverables for each development phase are standard in DoD acquisition. It is important that these deliverables are timed appropriately, tailored to the program's requirements, complexity, and risk, and balanced against the cost to deliver them. The DoD is currently piloting new processes on several acquisition programs in which acquisition oversight is altered in accordance with requisite program risk. This risk-balanced acquisition oversight is seeing some positive results, and is breaking new ground in rewarding acquisition capability improvement while maintaining realistic contractor expectations.

We strongly believe that integration should be added as one of the development phases. The report's focus is on weapon systems, where software development is truly embedded into larger systems. Integration has become one of the most critical activities in system development, given its impact and complexity increase with the size and complexity of the total system, or system of systems. This includes the integration of multiple software modules in development, integration of developed software with COTS or existing software, integration of hardware and software components, and integration of the software with other systems. DoD's move to acquire systems within a capabilities context that are interoperable and supportive of network centric enterprise services makes integration a key risk area that must be managed to achieve success.

**RECOMMENDATION 3:** The GAO recommended that the Secretary of Defense ensure that DoD has the knowledge it needs to oversee software intensive acquisitions by requiring software contractors to collect and report metrics related to cost, schedule, size, requirements, tests, defects, and quality to program offices on a monthly basis and before program milestones; require contractors to have an earned value management system that reports cost and schedule information at a level of work that provides information specific to software development. (p. 32/GAO Draft Report)

**DOD RESPONSE:** Partial Concur. We believe the report's third recommendation should be modified to read that "...the Secretary of Defense ensure that DoD has the knowledge it needs to oversee software intensive acquisitions by requiring software *developers* to collect and report metrics related to cost, schedule, size, requirements, tests, defects, and quality to program offices *in support of program decisions. Commensurate with each program's complexity and risk, the DoD should also require developers* to implement earned value management systems to report cost and schedule information specific to software development *at the configuration item level.*

This recommendation increases DoD insight into the developer's processes, but the implementation, and associated costs need to be balanced with the program's requirements, complexity and level of risk. Further, we have seen that obtaining data from the contractor does not provide the "knowledge" that DoD requires to oversee software intensive acquisition as stated in the recommendation. Experienced and trained software acquisition professionals possess the knowledge to oversee a software intensive acquisition. Without this, data collected from the contractor is irrelevant.

**RECOMMENDATION 4:** These practices should be included and enforced with controls and incentives in DOD's acquisitions policy, software acquisition improvement plans and development contracts.

**DOD RESPONSE:** This final statement in the recommendations section of the report discusses establishing control and incentives in policy. The Department's policy already addresses these issues as exemplified in the following table.

| | |
|---|---|
| DoD 5000.1, paragraph E1.25 | "Software Intensive Systems. Acquisition of software intensive systems shall use process improvement and performance measures. Selection of sources shall include consideration of product maturity and past performance." |
| DoD 5000.1, paragraph 4.3.2 | "Responsiveness. Advanced technology shall be integrated into producible systems and deployed in the shortest time practicable. Approved, time-phased capability needs matched with available technology and resources enable evolutionary acquisition strategies. Evolutionary acquisition strategies are the preferred approach to satisfying operational needs. Spiral development is the preferred process for executing such strategies." |
| DoD 5000.1, paragraph 4.3.4 | "Discipline. PMs shall manage programs consistent with statute and the regulatory requirements specified in this Directive and in reference (b). Every PM shall establish program goals for the minimum number of cost, schedule, and performance parameters that describe the program over its life cycle. Approved program baseline parameters shall serve as control objectives. PMs shall identify deviations from approved acquisition program baseline parameters and exit criteria." |
| DoDI 5000.2, paragraph 3.7.2 | "Entrance Criteria. Entrance into this [SDD] phase depends on technology maturity (including software), approved requirements, and funding. Unless some other factor is overriding in its impact, the maturity of the technology shall determine the path to be followed. Programs that enter the acquisition process at Milestone B shall have an ICD that provides the context in which the capability was determined and approved, and a CDD that describes specific program requirements." |
| DoDI 5000.2, Appendix E, Table E3.T3 | Software Resources Data Report. All major contracts and subcontracts, regardless of contract type, for contractors developing/producing software elements within ACAT I and ACAT IA programs for any software development element with a projected software effort greater than $25M (FY 2002 constant dollars). Submit data on each software element at the following times:<br>-180 days prior to contract award<br>-60 days after contract award |

| | |
|---|---|
| | -60 days after start of subsequent software releases<br>-within 120 days after software release or final delivery |

The report acknowledges the DoD's actions to incorporate systems engineering directives into policy. OSD oversight of service and agency implementation of Section 804 will provide additional controls as necessary.

The DoD recognizes the importance of good software development practices, and has undertaken many successful initiatives that are also acknowledged in the report. DoD strongly believes that software success is fundamentally dependent upon the greater systems engineering success. It is here where our attention, focus, and improvement can make the greatest contribution to achieving successful system acquisitions.

# Appendix II: Software Models

## Software Development

The Capability Maturity Model for Software (SW-CMM)[1]® describes the principles and practices underlying software process maturity and is intended to help software organizations improve the maturity of their software process in terms of an evolutionary path organized into five maturity levels. Except for level 1, each maturity level is decomposed into several key process areas that indicate the areas that an organization should focus on to improve its software process. Table 3 describes the characteristics of each level of process maturity and the applicable key process areas.

**Table 3: Highlights of SW-CMM**

| Level | Characteristics | Key process areas |
|---|---|---|
| 1 Initial | The software process is ad hoc, and occasionally chaotic. Few processes are defined, and success depends on individual effort. | |
| 2 Repeatable | Basic project management processes are established to track cost, schedule, and functionality. The necessary process discipline is in place to repeat earlier successes on projects with similar applications. | • Requirements Management<br>• Software Project Planning<br>• Software Project Tracking & Oversight<br>• Software Subcontract Management<br>• Software Quality Assurance<br>• Software Configuration Management |
| 3 Defined | The software process for both management and engineering activities is documented, standardized, and integrated into a standard software process for the organization. All projects use an approved, tailored version of the organization's standard software process for developing and maintaining software. | • Organization Process Focus<br>• Organization Process Definition<br>• Training<br>• Integrated Software Management<br>• Software Product Engineering<br>• Intergroup Coordination<br>• Peer Reviews |
| 4 Managed | Detailed measures of the software process and product quality are collected. Both the software process and products are quantitatively understood and controlled. | • Quantitative Process Management<br>• Software Quality Management |
| 5 Optimizing | Continuous process improvement is enabled by quantitative feedback from the process and from plotting innovative ideas and technologies. | • Defect Prevention<br>• Technology Change Management<br>• Process Change Management |

Source: Software Engineering Institute, Carnegie Mellon University.

[1] CMM is registered in the U.S. Patent and Trademark Office by Carnegie Mellon University.

## Software Acquisition

The Software Acquisition Capability Maturity Model (SA-CMM)® is a model for benchmarking and improving the software acquisition process. The model follows the same architecture as SW-CMM® but with a unique emphasis on acquisition issues and the needs of individuals and groups who are planning and managing software acquisition efforts. Each maturity level indicates an acquisition process capability and has several Key Process Areas. Each area has goals and common features and organizational practices intended to institutionalize common practice.

**Table 4: Highlights of SA-CMM**

| Level | Focus | Key process areas |
|---|---|---|
| 1 Initial | Competent people and heroics | |
| 2 Repeatable | Basic Project Management | • Transition to Support<br>• Evaluation<br>• Contract Tracking and Oversight<br>• Project Management<br>• Requirements Development and Management<br>• Solicitation<br>• Software Acquisition Planning |
| 3 Defined | Process Standardization | • Training Program<br>• Acquisition Risk Management<br>• Contract Performance Management<br>• Project Performance Management<br>• Process Definition and Maintenance |
| 4 Quantitative | Quantitative Management | • Quantitative Acquisition Management<br>• Quantitative Process Management |
| 5 Optimizing | Continuous Process Improvement | • Acquisition Innovation Management<br>• Continuous Process Improvement |

Source: Software Engineering Institute, Carnegie Mellon University.

## Integrated Model

In 1997 a team led by DOD, in conjunction with Software Engineering Institute, government, and industry, concentrated on developing an integrated framework for maturity models and associated products. The result was the Capability Maturity Model Integration (CMMI)®,[2] which is intended to provide guidance for improving an organization's processes and the ability to manage the development, acquisition, and maintenance of products and services while reducing the redundancy and inconsistency caused by using stand-alone models. CMMI® combines earlier models from Software Engineering Institute and the Electronic Industries Alliance into a single model for use by organizations pursuing enterprise-wide process improvement. Ultimately, CMMI® is to replace the models that have been its starting point.

Many integrated models consist of disciplines selected according to individual business needs. Models can include systems engineering, software engineering, integrated product and process development, and supplier sourcing. There are also two representations of each CMMI® model: staged and continuous. A representation reflects the organization, use, and presentation of model elements. Table 5 shows the CMMI® model for staged groupings.

[2] CMMI is registered with the U.S. Patent and Trademark Office by Carnegie Mellon University.

**Table 5: Highlights of CMMI Model**

| Staged grouping | Process area |
|---|---|
| Maturity Level 2 | • Requirements Management<br>• Project Planning<br>• Project Monitoring and Control<br>• Supplier Agreement Management<br>• Measurement and Analysis<br>• Process and Product Quality Assurance<br>• Configuration Management |
| Maturity Level 3 | • Requirements Development<br>• Technical Solution<br>• Product Integration<br>• Verification<br>• Validation<br>• Organizational Process Focus<br>• Organizational Process Definition<br>• Organizational Training<br>• Integrated Project Management<br>• Risk Management<br>• Integrated Teaming<br>• Integrated Supplier Management<br>• Decision Analysis and Resolution<br>• Organizational Environment for Integration |
| Maturity Level 4 | • Organizational Process Performance<br>• Quantitative Project Management |
| Maturity Level 5 | • Organizational Innovation and Deployment<br>• Causal Analysis and Resolution |

Source: Software Engineering Institute, Carnegie Mellon University.

# Appendix III: Section 804. Improvement of Software Acquisition Processes

(a) Establishment of Programs—

(1) The Secretary of each military department shall establish a program to improve the software acquisition processes of that military department.

(2) The head of each Defense Agency that manages a major defense acquisition program with a substantial software component shall establish a program to improve the software acquisition processes of that Defense Agency.

(3) The programs required by this subsection shall be established not later than 120 days after the date of the enactment of this Act.

(b) Program Requirements.—A program to improve software acquisition processes under this section shall, at a minimum, include the following:

(1) A documented process for software acquisition planning, requirements development and management, project management and oversight, and risk management.

(2) Efforts to develop appropriate metrics for performance measurement and continual process improvement.

(3) A process to ensure that key program personnel have an appropriate level of experience or training in software acquisition.

(4) A process to ensure that each military department and Defense Agency implements and adheres to established processes and requirements relating to the acquisition of software.

(c) Department of Defense Guidance—The Assistant Secretary of Defense for Command, Control, Communications, and Intelligence, in consultation with the Under Secretary of Defense for Acquisition, Technology, and Logistics, shall—

(1) prescribe uniformly applicable guidance for the administration of all of the programs established under subsection (a) and take such actions as are necessary to ensure that the military departments and Defense Agencies comply with the guidance; and

(2) assist the Secretaries of the military departments and the heads of the Defense Agencies to carry out such programs effectively by—

(A) ensuring that the criteria applicable to the selection of sources provides added emphasis on past performance of potential sources, as well as on the

maturity of the software products offered by the potential sources; and

(B) identifying, and serving as a clearinghouse for information regarding, best practices in software development and acquisition in both the public and private sectors.

(d) Definitions—In this section:

(1) The term "Defense Agency" has the meaning given the term in section 101(a)(11) of title 10, United States Code.

(2) The term "major defense acquisition program" has the meaning given such term in section 139(a)(2)(B) of title 10, United States Code.

# Related GAO Products

*Defense Acquisitions: DOD's Revised Policy Emphasizes Best Practices, but More Controls Are Needed*. GAO-04-53. Washington, D.C.: November 10, 2003.

*Best Practices: Setting Requirements Differently Could Reduce Weapon Systems' Total Ownership Costs*. GAO-03-57. Washington, D.C.: February 11, 2003.

*Best Practices: Capturing Design and Manufacturing Knowledge Early Improves Acquisition Outcomes*. GAO-02-701. Washington, D.C.: July 15, 2002.

*Defense Acquisitions: DOD Faces Challenges in Implementing Best Practices*. GAO-02-469T. Washington, D.C.: February 27, 2002.

*DOD Information Technology: Software and Systems Process Improvement Programs Vary in Use of Best Practices*. GAO-01-116. Washington, D.C.: March 30, 2001.

*Best Practices: Better Matching of Needs and Resources Will Lead to Better Weapon System Outcomes*. GAO-01-288. Washington, D.C.: March 8, 2001.

*Best Practices: A More Constructive Test Approach Is Key to Better Weapon System Outcomes*. GAO/NSIAD-00-199. Washington, D.C.: July 31, 2000.

*Defense Acquisition: Employing Best Practices Can Shape Better Weapon System Decisions*. GAO/T-NSIAD-00-137. Washington, D.C.: April 26, 2000.

*Best Practices: DOD Training Can Do More to Help Weapon System Program Implement Best Practices*. GAO/NSIAD-99-206. Washington, D.C.: August 16, 1999.

*Best Practices: Better Management of Technology Development Can Improve Weapon System Outcomes*. GAO/NSIAD-99-162. Washington, D.C.: July 30, 1999.

*Defense Acquisitions: Best Commercial Practices Can Improve Program Outcomes*. GAO/T-NSIAD-99-116. Washington, D.C.: March 17, 1999.

*Defense Acquisition: Improved Program Outcomes Are Possible*. GAO/T-NSIAD-98-123. Washington, D.C.: March 17, 1998.

*Best Practices: DOD Can Help Suppliers Contribute More to Weapon System Programs*. GAO/NSIAD-98-87. Washington, D.C.: March 17, 1998.

*Best Practices: Successful Application to Weapon Acquisition Requires Changes in DOD's Environment*. GAO/NSIAD-98-56. Washington, D.C.: February 24, 1998.

*Best Practices: Commercial Quality Assurance Practices Offer Improvements for DOD*. GAO/NSIAD-96-162. Washington, D.C.: August 26, 1996.

## GAO's Mission

The General Accounting Office, the audit, evaluation and investigative arm of Congress, exists to support Congress in meeting its constitutional responsibilities and to help improve the performance and accountability of the federal government for the American people. GAO examines the use of public funds; evaluates federal programs and policies; and provides analyses, recommendations, and other assistance to help Congress make informed oversight, policy, and funding decisions. GAO's commitment to good government is reflected in its core values of accountability, integrity, and reliability.

## Obtaining Copies of GAO Reports and Testimony

The fastest and easiest way to obtain copies of GAO documents at no cost is through the Internet. GAO's Web site (www.gao.gov) contains abstracts and full-text files of current reports and testimony and an expanding archive of older products. The Web site features a search engine to help you locate documents using key words and phrases. You can print these documents in their entirety, including charts and other graphics.

Each day, GAO issues a list of newly released reports, testimony, and correspondence. GAO posts this list, known as "Today's Reports," on its Web site daily. The list contains links to the full-text document files. To have GAO e-mail this list to you every afternoon, go to www.gao.gov and select "Subscribe to e-mail alerts" under the "Order GAO Products" heading.

## Order by Mail or Phone

The first copy of each printed report is free. Additional copies are $2 each. A check or money order should be made out to the Superintendent of Documents. GAO also accepts VISA and Mastercard. Orders for 100 or more copies mailed to a single address are discounted 25 percent. Orders should be sent to:

U.S. General Accounting Office
441 G Street NW, Room LM
Washington, D.C. 20548

To order by Phone:
Voice: (202) 512-6000
TDD: (202) 512-2537
Fax: (202) 512-6061

## To Report Fraud, Waste, and Abuse in Federal Programs

Contact:

Web site: www.gao.gov/fraudnet/fraudnet.htm
E-mail: fraudnet@gao.gov
Automated answering system: (800) 424-5454 or (202) 512-7470

## Public Affairs

Jeff Nelligan, Managing Director, NelliganJ@gao.gov (202) 512-4800
U.S. General Accounting Office, 441 G Street NW, Room 7149
Washington, D.C. 20548