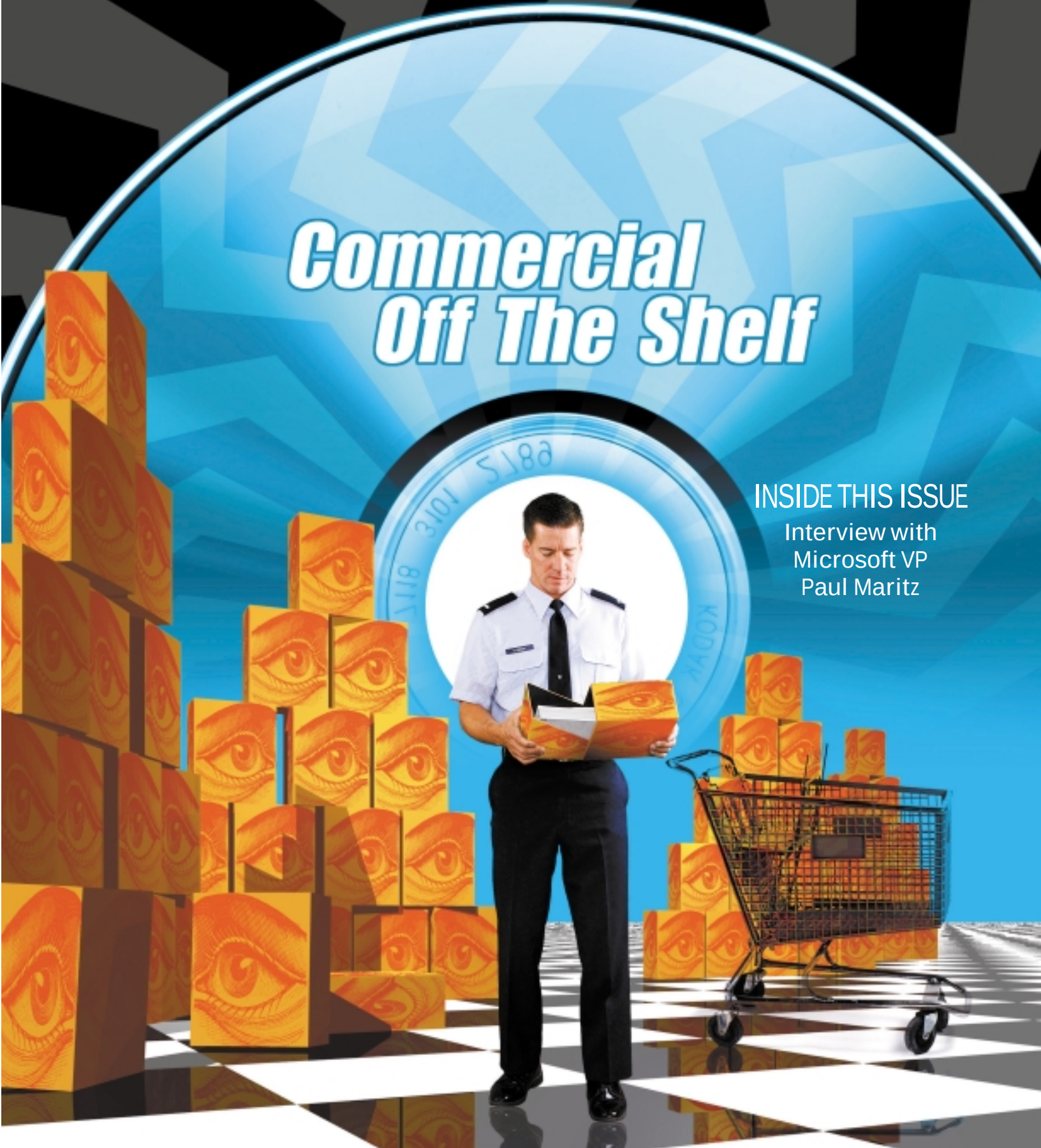
CrossTalk
September 2000 The Journal of Defense Software Engineering Vol. 13 No. 9

# Commercial Off The Shelf



## INSIDE THIS ISSUE

Interview with Microsoft VP Paul Maritz

## COTS



## Field Report



## Software Engineering Technology




On the Cover:
Kent Bingham, Digital Illustration and Design, is a self-taught graphic artist/designer and freelances both print and Web design projects. His portfolio is at www.adobe.com/eportfolio/kentbingham [Thanks to 1st Lt. Clint Stinson for being the model].

## Departments



CROSSTALK would like to welcome new Managing Editor Pam Bowers, who replaces Kathy Gurchiek.


## CrossTalk

SPONSOR *Lt. Col. Glenn A. Palmer*
PUBLISHER *Reuel S. Alder*
ASSOCIATE PUBLISHER *Lynn P. Silver*
MANAGING EDITOR *Pam Bowers*
ASSOCIATE EDITOR/LAYOUT *Matthew Welker*
ASSOCIATE EDITOR/FEATURES *Heather Winward*
GRAPHIC DESIGNER *Abby Hall*

VOICE 801-586-0095
FAX 801-777-5633
E-MAIL crosstalk.staff@hill.af.mil
CROSSTALK ONLINE www.stsc.hill.af.mil/Crosstalk/crostalk.html
CRSIP ONLINE www.crsip.hill.af.mil

***Subscriptions***: Send correspondence concerning subscriptions and changes of address to the following address. You may use the form on page 31.
Ogden ALC/TISE
5851 F Ave., Bldg 849, Rm B-04
Hill AFB, Utah 84056-5713

***Article Submissions***: We welcome articles of interest to the defense software community. Articles must be approved by the CROSSTALK editorial board prior to publication. Please follow the *Guidelines for CROSSTALK Authors*, available upon request. We do not pay for submissions. Articles published in CROSSTALK remain the property of the authors and may be submitted to other publications.

***Reprints and Permissions:*** Requests for reprints must be requested from the author or the copyright holder. Please coordinate your request with CROSSTALK.

***Trademarks and Endorsements:*** This DoD journal is an authorized publication for members of the Department of Defense. Contents of CROSSTALK are not necessarily the official views of, or endorsed by, the government, the Department of Defense, or the Software Technology Support Center. All product names referenced in this issue are trademarks of their companies.

***Coming Events***: We often list conferences, seminars, symposiums, etc., that are of interest to our readers. There is no fee for this service, but we must receive the information at least 90 days before registration. Send an announcement to the CROSSTALK Editorial Department.

***STSC Online Services:*** at www.stsc.hill.af.mil. Call 801-777-7026, e-mail: randy.schreifels@hill.af.mil.

***Back Issues Available:*** The STSC sometimes has extra copies of back issues of CROSSTALK available free of charge.

***The Software Technology Support Center*** was established at Ogden Air Logistics Center (AFMC) by Headquarters U.S. Air Force to help Air Force software organizations identify, evaluate, and adopt technologies to improve the quality of their software products, efficiency in producing them, and their ability to accurately predict the cost and schedule of their delivery.

# COTS: The Ideal World

Sit back, relax, and enjoy a ride in the ideal world as seen through the eyes of a software engineer. You wake up at 9 a.m. and get ready for your three hours of work. Breakfast consists of ham, eggs, toast, and a glass of milk; a breakfast that has no fats, no cholesterol, and will help you live another 200 years. You step to your spaceport and fly your car to work. When you arrive, your car folds into your briefcase. Finally, you are ready for three hours of monitoring the company computer as you wonder how your grandfather worked an eight-hour workday. And of course, every organization is CMM Level 5.

Sounds too good to be true, right? I consider commercial off-the-shelf (COTS) products to be like the ideal world for many system program offices and program managers; however, this is not the real world and COTS is not the silver bullet for software. Paul Maritz's interview on page 4 explains how Microsoft evaluates, selects, and integrates COTS software into its process. He states that it is important for Microsoft and their customers to use "standardized building blocks" for the products that they use and sell. However, any cost savings with using an "off-the shelf" product can easily be evaporated when layers of customization are included. He also gives some lessons learned to those interested in using COTS, especially to those in the government. Make the decision to use COTS a "top-down decision" so that the whole organization is using the same basic products, he advises, "Be willing to re-engineer your processes, if need be, to fit within the bounds of the capabilities of the software you are buying."

Fortunately, the government has seen some benefit in using COTS-based products, and is using Carnegie Mellon University's Software Engineering Institute to study ways in developing and supporting COTS-based systems (CBS). Lisa Brownsword, Patricia Obendorf and Carol A. Sledge's article, *An Activity Framework for COTS Based Systems* on page 8 summarizes the essential factors that distinguish CBS and describes a preliminary framework that captures new and changed activities necessary for a CBS approach and describes a suitable migration path.

Unfortunately as government agencies start using COTS and government off-the-shelf (GOTS) products, the program managers must realize the insufficient information and usage on the products' capability. Dr. William H. Dashiell and Phil Brashear's article, *Evaluating COTS/ GOTS Software: Functional Test Criteria* on page 17 paints a scenario between a program manager and a testing expert and how they determine the importance of developing and writing testable requirements for the program.

Finally, this issue of CROSSTALK should enlighten readers of the shortfalls and benefits of using a COTS-based system for software. As the Department of Defense idealistically prepares for the future, it must come to understand the best way to travel there and how to make the pieces fit.

Lynn P. Silver
Associate Publisher

CROSSTALK welcomes **Lt. Col. Glenn Palmer,** Director of the Computer Resource Support Improvement Program (CRSIP), Hill Air Force Base, and Utah. As CRSIP Director, Lt. Col. Palmer directs the transition and adoption of technologies by Air Force organizations to enhance their ability to acquire, develop, manage, and support mission-critical, software-intensive systems. Lt. Col. Palmer moves to Hill from Lockheed Martin in Ft. Worth, Texas where he served as Defense Contract Management Agency Program Integrator for the F-22 fighter program. He has 18 years experience in Air Force maintenance, engineering, and program management positions. He received his bachelor's degree in Mechanical Engineering from the University of Kentucky, a master's degree in Operations Research from the University of Northern Colorado, and a Masters of Business Administration from Creighton University.

# Up Close with Microsoft's Paul Maritz

CrossTalk had the opportunity to speak with **Paul Maritz** of Microsoft about commercial off-the-shelf products, open systems, quality, and Microsoft's software development culture. He is vice president of Microsoft's Developer Group, which includes platform technologies, development tools, and database products as well as providing support programs for the developer community. He is a member of the Microsoft Business Leadership Team, which shares responsibility with Microsoft CEO Bill Gates for the company's broad strategic and business planning. Maritz joined Microsoft in 1986 and has managed such software product groups as Networking, and Windows operating system and application units. He spent five years at Intel Corp. before joining Microsoft. He also worked for Burroughs Corp. and at the University of St. Andrews in Scotland. He is a graduate of the University of Cape Town and the University of Natal, South Africa where he studied computer science and mathematics.

CrossTalk: How does Microsoft determine whether to build or buy software?

**Maritz:** You have to decide first whether you want to do some unique things in the software. It is really a decision between what you want to achieve with the software; do you want to have a unique proposition for your customers? Secondly, if it is something you think is a standard piece you want to have inside your environment, are there such products available? Are they of sufficient quality? What are the business terms under which you can acquire them? All those things have to be thought through.

We do buy or license software in many cases. Over time, though, we tend to license them under scheme-ware by which we can put them through the same quality cycle, the same maintenance and support cycle. We are in the business ourselves of providing standard building blocks and our customers expect to see no difference between a piece of software that we built ourselves and one that we acquired. They do not want to be bothered by us saying 'we can support this one but we cannot support that one.' They want them to have the same level of quality.

We tend to look at situations where we have the opportunity to acquire software because in general it is a lot more efficient to acquire it because you do not have to do the development work yourself; however, you have to factor in:

- Do you want to have a unique value proposition?
- How are you going to support it?
- What quality level is it up to?
- The degree you want to integrate it into the rest of your environment.

In a lot of our cases you cannot buy an operating system with a full environment off the shelf, so we look to acquire elements of that and engineer it to certain quality levels and integrate it into the environment.

You cannot generalize too much in the decision criteria that we would go through as compared to a customer or user software. There is a different environment, although perhaps with some of the embedded systems or mission-critical systems in the Department of Defense (DoD) they would have to go through a similar set of criteria. You probably are going to have a differentiated competency or functionality vs. your enemy. It does not help you to have exactly the same software system as your enemy. You are going to want to make some changes to it, and you are going to want to customize it; you are going to want to engineer it to a certain level and support it over a period of time.

I think the first decision you have to make is what role is the software going to play in the environment? Then go from there.

CrossTalk: How does Microsoft evaluate, select, and integrate commercial off-the-shelf (COTS) software into its processes? Understand, this is a big issue in the military.

**Maritz:** We are in the business of trying to buy standardized building blocks that other people can use. While Microsoft will buy software for our products, we tend to buy them in the context of how we integrate them with the building blocks that we are going to supply. Another way you can look at it is how we use commercial software in the running of our company? We have made some very strong moves to get onto standardized software. Looking at our own internal financial and accounting software, if you go back five years, we had a hodgepodge of systems that we had taken and customized. We needed a good, efficient accounting system. We put ourselves through a very rigorous process of moving to the Systems, Applications, and Products in Data Processing (SAP) R/3 system; that was a three-year process, but it was a top-down decision. Today we are able to turn reports dramatically more quickly than in the past.

CrossTalk: That is very insightful. What you are saying is you do not create all of your own software.

**Maritz:** Absolutely. We want to focus our software development efforts on the things that will make us unique as a company and take effort off the things that will not make us unique. I think any entity has to think about those two things very carefully. The hard thing about getting in the areas that are not critical and [where] you do not have to be unique—it requires an explicit top-down decision. It really has to be a business decision; if you leave it to the technology people, there will always be 15 reasons why you cannot move.

CrossTalk: Do you have a percentage as to how much is COTS and how much is your own software that you use?

**Maritz:** Our software efforts fall into three categories:

**Areas where we decide to build software products—** operating systems, personal productivity tools, development tools. In those areas, we do buy the software; we tend to buy it and integrate it into our process. The goal is when it reaches the software it is indistinguishable from our other pieces of software.

**Areas where we act as a consumer of our own product—** wherever possible, we should be using the same products that we sell to our customers. We use Exchange as our messaging system; that was not always the case. There was a time when we had our own home-brewed, internal electronic mail system.

**Areas where we decide not to build software products—** there is a lot to be gained by leveraging the research and develop-

ment that SAP does across a variety of industries rather than trying to have unique software for our environment.

The hard thing is you have to strike the right balance between being willing to change your internal ways of working and processing, and getting the software customized. If you take the off-the-shelf product and decide that you are not going to change the way you work, you run the risk of writing a layer of customized software that puts you back in the situation where you were before—having unique, expensive software that you have to maintain.

You have to be willing when you go to a COTS environment, to look at your business processes and be willing to re-engineer them. Most companies that are really successful realize the value of that and are willing to do business process and re-engineering in parallel to putting the standardized software in.

There has to be a balance between getting the software to do what you want it to do and working within the bounds of the software. Otherwise you can quickly evaporate any cost savings.

CrossTalk: Do you have any lessons learned from working with COTS products?

**Maritz:** There are really two key lessons:

[1] You want to make it a top-down decision. Typically you want to get all of your organization to use the same basic, underlying products; you do not have multiple relationships, multiple learning. Approach it as an important philosophical or business strategy decision and be prepared to put the willpower and effort to see it all the way through.

[2] Being willing to re-engineer your processes, if need be, to fit in the bounds of capabilities of the software you are buying.

CrossTalk: The DoD is developing a common operating environment to improve interoperability. Microsoft seemed to develop a common operating environment in Windows for commercial use. Do you have any advice to the DoD in its endeavor?

**Maritz:** Be pragmatic about things. It is unlikely that the world is ever going to move to one operating environment. There are always going to be three or four commonly used operating environments. The critical thing for the DoD is to be able to leverage the research and development that went into each of those environments. If you get too far ahead of industry, what happens is that in order to get products for a common operating environment, you end up being the one who has to fund the new development for it. You can get in the situation where the DoD has to pay for vendors to adapt their products to a common operating environment, which is not a good situation to be in either.

Having standards is very important. You do not want to have whole parts of your organization going in different directions without being able to reuse and leverage your investment.

The other thing is to recognize that most important standards in the industry have evolved in a de facto way out of existing products, rather than in a top-down way. The real challenge is to have a balance of bottom-up and top-down, recognizing what is working in the industry and being willing to adapt your framework standards to take advantage of those.

Do not get caught in the trap of doing standards for standards' sake. Otherwise you could end up funding it all, which is not what you want to do. What you want to do is leverage other people's investment.

CrossTalk: If Apple OS9 represents a closed operating system and Linux represents an open operating system, where is Windows NT on this continuum?

**Maritz:** Is Linux an open operating system? You have free access to the source but is there any official standard party that controls the interfaces to Linux? No. Is that an open process or not? I do not know.

We do not think of [Windows NT] either as a closed or open operating system. We think of it as basically as a system designed to solve computing problems for a wide variety of users.

CrossTalk: How do you maintain interoperability among third-party application developers?

> "Great software design is being able to strike the right balance between the right granularity in terms of your design, and the practicality of really using your resources, CPU, and memory efficiently."

**Maritz:** We do this by maintaining a complete set of compatible Windows-based interfaces that we maintain on a generation-to-generation basis.

We are also very careful not to break compatibility when we release new versions of Windows. There are literally hundreds of man-years that go into preserving that application invested by third-party developers. We literally have to test thousands of applications to make sure that in each application, third-party investment by those who have written the applications, can be leveraged in going forward.

On the other hand, we also provide standard interfaces, such as the Internet set of interfaces (HTML, XML, etc.), that people have invested [in] that are in or around those interfaces.

We have a responsibility to the Windows developer community, which is a very large, very important community. And we have a responsibility to try and provide the other important standards that are in the industry.

CrossTalk: On your third-party developers, you do not set any kind of process standards or testing standards?

**Maritz:** We have to walk a fine line between not trying to tell other entities how they should run their businesses, and providing some guidance to customers as to what they can expect from an application. We do have a certification program where people can certify that the applications have a certain characteristic but we cannot be in the position of determining other companies' fate by blessing or not blessing their applications. We have to provide the tools that enable people to write good applications, but we cannot be the ones to be the arbiter of their business success.

CrossTalk: The SEI Capability Maturity Model (CMM®) has a strong influence on defense software development. Does Microsoft use the CMM? Why or why not?

**Maritz:** We do not use the CMM directly. We use common ideas from the CMM and again, this is an issue of looking at the different environments that we operate in. There are both similarities and differences when you are developing software basically as a component for some larger system. You are developing software that has to be sold as a product in millions of units. I think there

is a common element, which is discipline and maturity. Those are, without any question, key characteristics. We use different mechanisms to assure discipline and maturity in our process, which do not fit into exactly the same terminology or sequence that CMM does. But the goal is the same.

We have different issues that come into play. Typically in a CMM model, it is understood that the requirements are well understood up front. We operate in an environment where you know there are certain basic requirements you understand up front, but you really do not know what is going to be required until you can get the product in use by users. People are going to use this product in many, many different ways. We have to have a different methodology that gets user feedback much earlier into our cycle.

I think one of the major differences between our process and a traditional Waterfall software process is that we use what we call a milestone methodology. Essentially you release the product in a series of releases; typically, anywhere from two to four releases occur. You try to build a base set of functionality and try to subset that functionality so you can get it useable by a certain critical mass of users as early as possible in the cycle. That is important for two reasons:

[1] You cannot anticipate how people might react to the product. It is very important how the end users are going to react. Are the dogs going to eat the dog food?

[2] No matter how much effort we put into writing software test suites—we literally have one developer writing test code for every developer writing code to begin with—you cannot predict all the ways the product is going to be used. The product is always going to be used in unexpected ways that are not covered by your test suites. It is not enough to have a product that passes your test suites. This is something we learned in the 1980s, together with IBM, when we were applying traditional software methodologies to developing commercial software. We got in situations where we would parcel our test suites and release the product and people would say, 'that product is terrible.' You have to strike a balance between different techniques for assuring quality. Quality comes from design discipline and formal software verification via test suites, and actual product usage. It is really the union of those things that we have found [is] what gives a good product.

We have a different way of trying to inject discipline and maturity into our process, but the goal is the same. It takes time. We found it takes awhile for a team to develop that can both formally meet the criteria and informally function well enough as a team to deliver on that.

CrossTalk: How does Microsoft assure software quality?

**Maritz:** We try [to] put a net together of well thought-through designs, and use design quality checks such as design walk-throughs, peer code reviews, etc.

We write an extensive amount of software that we use to test our own software. In addition to that, one of the things we have found over the years developing very large pieces of software is that you have to be constantly measuring yourself as to where you really are. Also, [we follow] this notion of exposing yourself early to end-user testing.

We typically look at a software project and we have found there are several questions you can ask a team. They will sure tell you whether they are out of control and have bad quality. How regularly is the whole team putting the software together? What we have found is that no matter how good a design is and how well you have decomposed the problem, nobody is ever perfect in their design and how all the pieces are going to splice together. If you have a team that is not mature, and not functioning well, it tends to leave that very late in the cycle. It is a real effort to put all the pieces together and integrate them on a regular basis.

One of the disciplines we use is insisting on regular integrations. If a team cannot integrate all its pieces, on a weekly basis, that is a warning signal. It means it really does not have the right contract between design elements in place. Because our software methodologies are good at describing structural aspects of software, we do not have good theoretical tools for predicting performance in terms of software design. Unless a team has written tests to measure its performance as being rigorous about measuring itself and performance, that can be a warning signal. In many cases, having too many layers and too many interfaces militates against performance.

Great software design is being able to strike the right balance between the right granularity in terms of your design, and the practicality of really using your resources, CPU, and memory efficiently. Forcing yourself to be honest about where you are in terms of performance is another key issue.

Another issue is your bug backlog. No team can recover if it lets its bug backlog, its performance backlog, or its design backlog get too far out of hand. What we try and do is structure our life cycle to say you really are going to release this product three or four times and you need to have an audience and a set of criteria of quality and performance associated with each of those releases.

You have got to be very disciplined about treating each of them as a release. When you hit a milestone, you have got to say 'we have formally met our quality criteria, we have formally met our performance criteria, and here is this set of people, outside of the team, that says 'Yes, this is usable, you are on the right track.'

One of the key things important for people to internalize is that quality is an absence of bugs, it is the right performance, and it is the right set of features. Anyone of those three things can lead a product to have a bad reputation.

CrossTalk: What standards are used by Microsoft to assure product quality? How are they created? How are they enforced?

**Maritz:** The key is to get people to be honest in articulating up front the criteria they are going to use. Realize that you also have to revisit your criteria at each milestone, because you will learn when you reach a milestone about the nature of the software you are building. It is OK to revise your criteria, but only when you get to a milestone. When you get to a milestone, there are two things you learn—if you met your original set of criteria, and if you should revise your criteria for the next milestone?

CrossTalk: But is that difficult to do? Change is always difficult, you have a process and sometimes you get so wedded to the process.

**Maritz:** Exactly. Often there is a lot of pride involved and

people like to kind of fudge on the milestones. That tends to be pennywise, pound-foolish. You are much better off forcing issues out earlier in the cycle.

CrossTalk: What is the single most important aspect to assuring the quality of Microsoft software?

**Maritz:** It is hard to say the single most important aspect. We try to use a network of different techniques and ways of looking at the problem. You cannot simply say formal methods alone will do it for you. Formal methods will get you some of the way. However, informal methods tend to be very important because they ultimately tell about you whether it is the right product or not, but they can be very difficult and expensive to use.

It is basically a toolbox of techniques that we have learned over the years. The thing that is most emblematic of our process is this notion of milestones. You try and decompose the product —not just breaking it into subsystems, but equally important, breaking it into releases where each release has been chosen to have a certain level of functionality that you can get objective, formal testing on and objective user-testing on. Both are needed to prevent situations where you get products that go out of control and you find out too late in the cycle.

CrossTalk: How does Microsoft attract and retain good software developers and managers?

**Maritz:** With difficulty, like everyone else. You try and create a culture where those people believe they are valued. You try and create a culture where they believe they can do interesting and important work. You balance that with the fact that you cannot be doing interesting and important work all the time. There has to be a balance between inspiration and creativity, and plain old hard work and slugging through it.

We try to create an environment where software engineers know they are at the core of what we do. They are valued as individuals. If they put in hard work they will be rewarded by being treated well, being compensated well, and getting the opportunity to do interesting work. They can continue to grow as individuals.

Most people who do software development do it only partly as a way of earning a living. They do it because they have a passion for it and they get a lot of satisfaction out of doing it.

CrossTalk: It is something that the DoD struggles with, attracting and retaining that intellectual resource.

**Maritz:** Are you putting enough attention into structuring so that these people have an interesting career path and get to work on interesting problems? I think there should be no shortage of [interesting software problems] in the Department of Defense, balancing that with the fact that every job in life has a certain amount of plain old hard work associated with it.

The world is becoming a software world. There is no aspect of life that does not have software in it these days. Consequently, there is tremendous demand for these people. These people are a scarce resource and have to be compensated accordingly.

CrossTalk: Do hotshot software engineers make good software managers?

**Maritz:** Not necessarily, but some of them do. There are certain hotshot software engineers who have no interest in being a manager, will be bad managers, and you have to construct a career path so those people can advance and be valued by the organization without having to become a manager. In many cases that means teaming them up with somebody who is a good manager.

We have a distinguished engineer program, where somebody can rise up where essentially they are compensated as a senior executive of the company would be compensated.

You need both. You need great architects and great software managers. Finding those people who have the right characteristics —their level of maturity and their propensity to want to work with other people—and encouraging them is very important.

The best software managers are those who also have a good understanding of what their people are doing. They do not have to be expert, but they have to have a good appreciation for what people are doing. Software engineers do not have a lot of respect for their leadership unless they know their leadership has a basic understanding of what their issues are and have sympathy for the challenges that they face. [Knowing] if some critical decision has to be made that that decision will not be made randomly. The manager will have enough background to know whom to ask and get good advice.

The key is to get the individual to recognize which track he is on. It is, in many cases, convincing people who are on a technical track or management track to be comfortable with which track they are on, accept that, and realize that they do not have to be the other in order to get where they want to go.

CrossTalk: What qualities that are rare among software engineers do you look for in a good software manager?

**Maritz:** When you become a manager, you can never be the expert in all areas. You are somebody who has to show leadership by absorbing a lot of good ideas and advice from other people. Those people who tend to do that are more secure in themselves. They are able to take feedback. They are not afraid to appear to be dumb by asking questions and learning. They tend to be people who can get honest feedback. They are people who have a lot of self-confidence with a small 's' and a small 'c.' I'm not talking about braggadocio. And [they] also have a good appreciation for what it is the team is trying to do. You are leading technical people. You have to make it a point to know the problems that people are facing. You do not have to come up with the answers, but you have to know enough to realize when somebody is trying to put one over on you, when somebody is lying, or when people are in need of help.

CrossTalk: What does Microsoft do to encourage long-term employment and loyalty from productive software developers?

**Maritz:** It is a combination of three things:

[1] Making them feel very good about working in an environment that values and respects them—they feel they are *mission critical.* Nobody wants to work long-term for an organization where you think you are in a secondary or under-appreciated role.

[2] Making sure that they can work on things that engage them—every job has its element of tedium associated with it, but you have to make sure there are enough interesting and exciting things that people can feel engaged and challenged.

[3] Compensating them.◆

# An Activity Framework for COTS-Based Systems

Lisa Brownsword, Patricia Oberndorf, and Carol A. Sledge
*Software Engineering Institute, Carnegie Mellon University*

*As use of commercial technology and products in systems becomes increasingly popular, particularly for government organizations, a new understanding of the dynamic principles of system creation is needed. However, there is little information on how using commercial off-the-shelf (COTS) products affects existing system development practices or what new processes are needed to be successful with COTS products. As part of the COTS-Based Systems Initiative at Carnegie Mellon University's Software Engineering Institute (SEI), we are studying this new software development process and have started to articulate some of the activities and practices that can be followed for development and lifetime support of COTS-based systems.*

Many government and industry programs are discovering difficulties that can accrue in creating a COTS-based system [1, 2, 3]. Some are starting to describe new processes and procedures [4, 5]. But none of this work brings together a description of all of the activities—engineering, business, contractual, and management—that are new or changed as a result of using COTS products and technology.

The basis of our work is more than 30 medium and large projects, ranging from business information management systems to embedded weapon and military command and control systems. We have captured information about the practices that did and did not work. This information has been analyzed to understand and characterize the common points of success and failure. This resulted in identifying the process changes required to address these real-life lessons and the articulation of a framework for organizing the new and changed process elements.

In this article, we will summarize the essential factors that distinguish COTS-based systems (CBS) and describe a preliminary framework that captures new and changed activities necessary for a CBS approach. It is drawn from the collective experience of the members of the CBS Initiative at the SEI and studies undertaken in the context of working with individual organizations on their systems [6], [7]. In many instances the strawman framework describes parts of approaches that have been used by organizations. To date no one organization has consciously pursued their work according to this set of ideas. The framework and its contents can be used by projects in several ways: to determine what practices are required for effective leverage of the COTS marketplace, to identify the difference between their existing practices and those required, or to determine a suitable migration path.

## CBS Process Drivers

Many software practitioners today are unfamiliar with CBS development, potentially involving a dozen or more COTS products[1] and custom or legacy components that provide system functionality. A philosophy and process that is different from that familiar to custom developers is needed for COTS-based systems. New process drivers flow from the definition of a "COTS product" and from the consequences of assembling things from purchased parts:

- CBS development is an act of composition.
- CBS development is shaped by realities of the COTS marketplace.
- CBS development occurs through the simultaneous definition and tradeoffs of the COTS marketplace, your architecture, and your system context.[2]

### CBS Development as Act of Composition

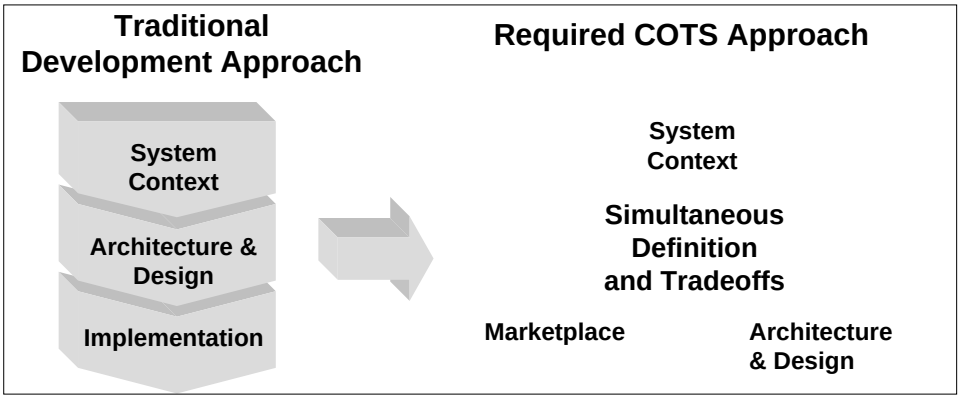
The first driver holds that the development of a custom system is essentially an act of creation, whereas the development of a COTS-based system is ultimately an act of composition and reconciliation. Custom development starts with the system requirements and creates a system that meets them—we are producers. However, CBS development starts with a general set of requirements and then explores the offerings of the marketplace to see how closely they match the needs—we are consumers who must integrate the products we buy into a system that meets the need. The nature, timing, and order of activities done and the processes used differ accordingly.

### CBS Development Shaped by Realities of COTS Marketplace

Eight inherent marketplace characteristics help determine the nature and evolution of a COTS-based system endeavor:

- There is frequent, continuous change in COTS products and the marketplace.
- The marketplace, noth the needs of any particular system, drives COTS products.
- Products have built-in assumptions about how they will be used; these might not match the system users' processes, resulting in clashes.
- Licensing and data rights will affect cost, architecture, and data processes.
- Projects have limited control over a COTS product's release frequency or content.
- Projects have limited visibility into COTS products' source code and behavior.
- Products are built on architectural assumptions that can vary across system components and could conflict with an evolving system architecture.
- COTS products have interdependencies.

Figure 1. *Traditional vs. COTS-Based Approach*



### CBS Development Through Simultaneous Definition/Tradeoffs

The last driver of CBS processes is really a consequence of the previous two: there is a fundamental change required in the approach to system development for COTS-based systems, as pictured in Figure 1. On the left is a traditional custom-development approach in which requirements (referred to as system context) are identified, then an architecture defined, and finally (custom) implementation is undertaken.

But if this is applied to CBS, it is unlikely that the marketplace will yield any products that fit the *a priori* requirements and architecture. Instead, with CBS it is necessary to consider system context, architecture, and the marketplace *simultaneously,* as pictured on the right of Figure 1. Any of the three may have impacts on the other two; none can proceed without knowledge and accommodation of the other two. Further, the activities that are performed for CBS systems are cyclic in nature; these tradeoffs will be repeated frequently throughout the lifetime of the system. This fundamental change not only necessitates changes in the engineering processes but also in the processes used to acquire and manage the construction of such systems.

## CBS Activity Areas

To understand the process changes generated by the use of COTS products, we identify activities that are either new for COTS-based systems or were present in custom development but change for CBS development. These activities are grouped into four major *activity areas* in Figure 2: Engineering, Business, Programwide, and Contract. Engineering and Business activity areas are straightforward. The Contract activity area covers issues involved in contracting with vendors and integrators. The Programwide activity area accounts for activities that are not contained in one area but span multiple areas.

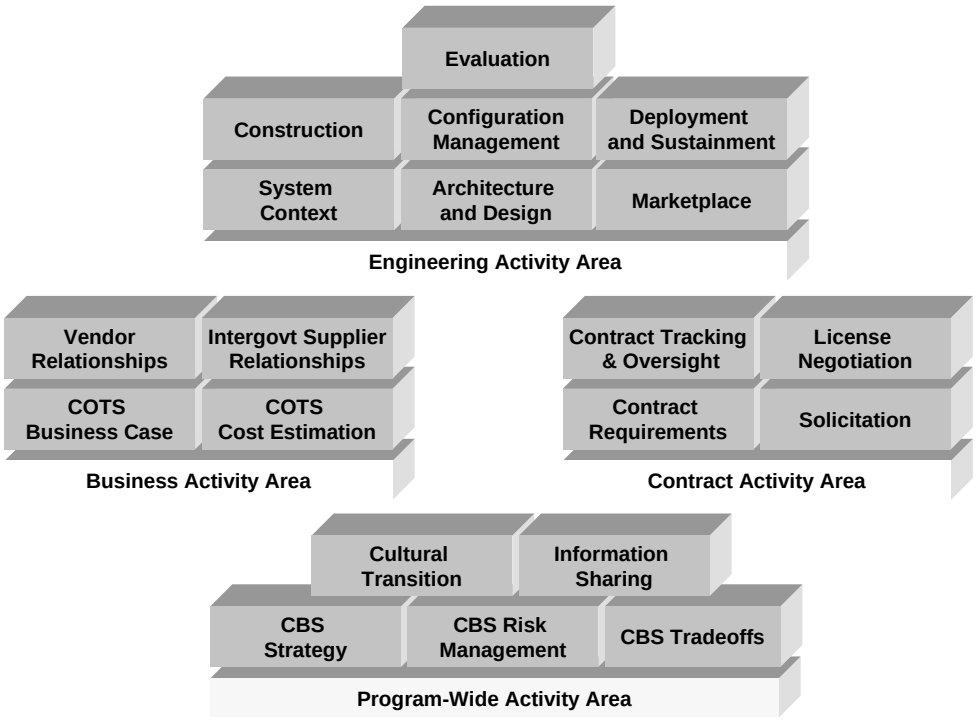
Within each of these activity areas, the new and changed activities are categorized into a number of *activity sets,* represented as blocks in Figure 2. Each set of activities operates continuously. There is no implied sequence within an activity area. Rather, the activity sets represent categories of related activities. For each activity set, we have identified its scope, the activities, and usage guidance or tips to consider in defining and applying the activities.

Our goal is to emphasize the differences from traditional custom development processes. Sometimes the differences are not in *what* is done but rather *how* or *when* or with what *marketplace considerations* the activity is done. For example, the activities in the CBS Risk Management activity set are the same activities used in any form of risk management. The difference derives from the nature of COTS risks associated with the use of COTS products that have not been encountered before and the diversity of the mitigations that are required.

The activity areas and their activity sets are a preliminary notional model that would be used to guide the detailed planning of a specific program. Depending on the particular needs of a program, some activity sets would have greater emphasis than others. The identified activity sets apply not only to new programs but also to existing programs.

In the sections that follow, each activity area is summarized indicating some implications of the CBS process drivers, some of the interrelationships among the activity sets, and a tabular illustration of an activity set to indicate the type of information we have gathered. A more complete treatment of each activity area and its associated activity sets is provided in [8].

Figure 2. *COTS-Based Systems Activity Areas*




### Engineering Activity Area

The Engineering Activity Area, as shown in Figure 2, is associated with the technical conceptualization, construction, and maintenance of a system. To a large extent these activity sets operationalize the CBS approach indicated on the right side of Figure 1. Activities in one set are done concurrently with and with mutual cognizance of other activity sets.

**System Context Activity Set** (an extract is shown in Table 1)**.** Mismatches between end-users' processes and the processes embodied in COTS products will occur, and these differences will constrain the system context and the program's ability to leverage the marketplace. Late discovery of these mismatches has been the foremost COTS issue for many projects we studied. Combined with other features of the CBS process drivers, these mismatches demand early and continual involvement of the program's stakeholders across all engineering activity sets. Their help is needed in deciding the potential tradeoffs between requirements and available COTS products and technologies; things move too fast to recover if their input is sought too late.

**Marketplace Activity Set.** COTS-based systems are by their nature highly evolutionary. This derives in part from the usual changes in end-user needs. But new CBS process drivers show that the

**System Context:** Encompasses considerations that define, prioritize, and constrain the CBS to be fielded, such as functional/nonfunctional requirements, end-user operations, business drivers, operational environment, and constraints such as schedule and budget.

| Activities | Tips |
|---|---|
| • Determine and prioritize the negotiable/non-negotiable elements of the system context.<br>• Understand the essential elements of the business processes and identify process/product mismatch before committing to the marketplace.<br>• Re-engineer business processes in light of knowledge of available products.<br>• Negotiate system context changes as part of COTS-based tradeoffs.<br>• Dynamically reflect results of COTS-based tradeoffs in the system context.<br>• Re-examine COTS products for opportunities to optimize user processes. | • Requirements analysis is still a necessary activity, although a new process is required to deal with simultaneous tradeoffs.<br>• Leverage prototypes to gain sufficient product insight to understand the extent of match with the end-user processes.<br>• End users tend to consider current processes and preferences to be non-negotiable.<br>• Engage all appropriate stakeholders early and often. |

Table 1. *Extract from System Context Activity Set*

**COTS Business Case:** Provides the basis for "make vs. buy" decisions for an entire system or an individual component. Covers the information gathering and analyses necessary to reach a recommendation regarding which of several alternative COTS or custom solutions to choose. Uses many of the other engineering and business activity sets.

**Activities**

- Determine CBS success factors.
- Conduct preliminary study of feasibility of a solution using COTS products.
- Identify key COTS-related assumptions.
- Articulate alternatives to be analyzed.
- Formulate CBS strategic plans.
- Analyze financial implications (costs, CBS risks, costs for risk mitigation).
- Analyze (COTS and non-COTS) alternatives and determine recommendation(s).
- Revisit the business case periodically and at key reassessment events.

**Tips**

- Information gathering is key. Beware of the very short half-life of the information due to marketplace volatility. Beware that comparisons are not usually of the "apples to apples" variety.

Table 2. *Extract from COTS Business Case Activity Set*

marketplace creates a new source of evolutionary demands, based both on the natural ebb and flow of products and technologies and on end-user discovery of new capabilities that have emerged in the marketplace.

**Architecture Activity Set.** The evolutionary nature of the marketplace has a particularly strong impact on the system architecture and design, as both must now be devised to withstand years, if not decades, of change. An architecture that allows a system to evolve efficiently is a strategic asset for CBS—it is the only thing the project owns.

**Construction Activity Set.** COTS product modification is often a temptation. Avoid it; if it cannot be avoided, successful projects go into it with a clear understanding of what it will mean in the future. "Modified COTS" is an oxymoron; once you have modified a COTS product for a specific use or system, it is no longer COTS. System lifetime costs of COTS product tailoring or modification must be taken into account as part of architectural and product selection decision-making.

**Configuration Management Activity Set.** Configuration management is still critical, but there are additional demands. Product versions and product dependencies on specific versions of other products must be tracked. License information and management (in the Contract activity area) may need to be accommodated as well.

**Deployment and Sustainment Activity Set.** The traditional separation of development and sustainment blur and become indistinguishable. Sustainment events, such as product upgrades, will occur before initial delivery of the system, and construction activities such as product selection, test, and integration will be necessary during sustainment. This impacts many other activities, such as budgeting, staffing, and contracting, and holds true from the purchase of the first product until retirement of the system.

**Evaluation Activity Set.** Evaluation of products and technologies begins from the moment the initial idea for a system is conceived, permeating and underlying all the other activities continuously throughout the CBS lifetime. This suggests dedicated evaluation resources (people, software, hardware, and facilities) as the useful half-life of market information is very short—about six months—and product information may not be valid for significantly longer periods of time.

## Business Activity Area

The Business Activity Area (refer to Figure 2) is associated with developing the business case and cost estimates, and managing vendor and other supplier relationships. COTS product and technology decisions are not just engineering decisions, they are also business decisions. Many activities in the business activity area require information from the engineering activity area and vice versa. For example, to create a COTS business case requires detailed COTS product information derived from the Marketplace and Evaluation activity sets plus architectural and design prototypes from the Architecture activity set.

COTS Business Case Activity Set (an extract is shown in Table 2). Making a COTS product purchasing decision in most cases is more than buying a commodity. A program's potential success may now be tied to a critical component from the vendor, requiring an effective long-term business relationship. The purpose of the business case is to analyze the alternatives to find the one with the best return on investment, total cost of ownership, and risk profile that best fits the requirements across the life of the system. Business decisions made regarding COTS products and technologies must incorporate the total cost of ownership across the system life, not just initial purchase costs. A key part of constructing the COTS business case is gathering information from such sources as market research and trend analysis, gap analysis between the product and your requirements, investigations of vendor health and practices, and detailed product usage through prototypes, demonstrations, and pilots.

**COTS Cost Estimation Activity Set.** The CBS process drivers are felt as strongly in the business activity area as in engineering. CBS cost estimation must account for all the differences for CBS implied by the CBS process drivers. In addition to many traditional costs, a CBS may incur costs for reacting to new product releases and marketplace changes (including "end-of-life" events), technology refresh, continuous evaluation, marketplace and technology watches, licensing, (re)integration, etc. Development of new publicly available COTS cost estimation techniques and models is in its infancy.

**Vendor Relationships Activity Set.** A vendor relationship is the means of partnering with a vendor that is important to your

system. The relationship relies on a cooperative exchange that explores current and future vendor and government plans. Such relationships provide insight into product releases, and represents a means for the government to influence vendor plans or directions. The type and depth of vendor relationship is dependent on the importance of the COTS product to the system and the importance of the program as a customer of the vendor.

**Intergovernmental Supplier Relationships Activity Set.** For DoD programs, another DoD organization may be the supplier of a nondevelopmental item that is incorporated into your system. The other DoD organization is not a vendor but is a supplier who exhibits both similarities as well as differences from how a commercial vendor would operate. A program needs to cultivate and nurture a relationship with these suppliers

## Contract Activity Area

The Contract Activity Area, indicated in Figure 2, is associated with the contractual aspects of creating and monitoring relationships with integration contractors and vendors. The first three activity sets focus on the contract relationship with an integration contractor. They emphasize the aspects that are different for a CBS contract, so they are not a complete guide to the contract effort for a CBS. The license negotiation activity set covers the contractual relationship with vendors.

**Contract Requirements Activity Set.** Contract requirements define the scope of the contract effort for CBS integration. They are developed with stakeholders, including potential bidders and suppliers. Contract requirements need to be flexible, traceable and verifiable, and address the system service lifetime. Contract requirements should accommodate CBS issues such as (but not limited to) technology refresh, version upgrade plans, market and technology watch groups, evolvable architecture, and supplier support.

**Solicitation Activity Set.** Look for CBS contractors who show successful previous CBS experience and knowledge of the COTS marketplace relevant to your domain. Consider carefully that their engineering and management practices provided in their proposals address the CBS issues and the risks specific to your program. Realistic demonstrations should be a key part of the contractor selection process in many situations.

**Contract Tracking and Oversight Activity Set.** Execution of activities for this activity set requires government skill and experience in such areas as COTS cost estimation, oversight of contractor's iterative development, relevant marketplace trends, evolving the system architecture, and product upgrades and technology refresh issues.

**License Negotiation Activity Set** (partial extract shown in Table 3). License agreements lay the foundation for and embody the program's vendor relationships. They must withstand many changes to a vendor's product and must be carefully considered. In particular, the program must be sensitive to their impact on program costs and potentially on the system architecture.

## Programwide Activity Area

The Programwide Activity Area (shown in Figure 2) covers the activities that span the Engineering, Business, and Contract Activity Areas in order to develop and sustain a CBS.

**CBS Strategy Activity Set** (extract shown in Table 4). The CBS strategy sets the stage for how a project will conduct all other activities. For example, the CBS strategy governs to what depth a COTS business case will be done, what investment will be made in vendor relationships, and what development approach will best support a CBS approach. Due to the continual changes in the COTS marketplace, a program will need to re-evaluate its CBS strategy periodically and adjust its plans and actions accordingly.

**CBS Risk Management Activity Set.** The goal of risk management for CBS is to identify COTS risks as early as possible, adjust the strategies and plans to manage those risks, and develop and implement a COTS risk management process as an integral part of an organization's overall CBS approach. Given marketplace volatility, COTS risks are likely to change more rapidly than the typical risks associated with custom systems. Examples of common COTS risks include a key vendor going out of business or an engineer's inability to integrate two selected products.

**CBS Tradeoffs Activity Set.** Engineering has always been an exercise in tradeoffs. With CBS, new tradeoff considerations arise, such as products that do not meet requirements, effects of licenses on design decisions, a vendor's or supplier's market share, architectural mismatch among components, long-term viability of a technology, product, or vendor, and the (mis)match of COTS product processes and existing end-user processes. Compounding the tradeoff issues is a program's lack

Table 3. *Extract from License Negotiation Activity Set*

**License Negotiation:** Looks at what the vendor offers with respect to terms, conditions, and costs for a given product for use in an organization over a particular period of time. Based on the situation and needs of the organization, negotiates the license(s) that best suits both parties.

| Activities | Tips |
|---|---|
| • Investigate licensing alternatives and costs; capitalize on enterprise licensing opportunities.<br>• Incorporate nonstandard provisions, such as vendor commitment to inclusion of modifications, integration support, notification of product splits, and license transfer.<br>• Negotiate licenses. | • License agreements may be used to describe the relationship with the vendor, incorporate nonstandard provisions such as vendor commitment, inclusion of modifications into the next commercial product release , and the kind and degree of integration support the vendor will provide. |

Table 4. *Extract from CBS Strategy Activity Set*

**CBS Strategy:** Seeks to derive an approach to COTS-based system development that will meet CBS objectives within program constraints over the life of the system. Provides for formulating, conducting and documenting planning activities for a CBS, providing one aspect of an overall system strategy. Factors in the realities about the COTS marketplace and the challenges that result into the system planning.

| Activities | Tips |
|---|---|
| • Identify CBS goals, constraints, and assumptions.<br>• Identify COTS-related risks.<br>• Identify relevant market segments.<br>• Identify alternative COTS-based solutions.<br>• Reassess CBS strategy as necessary.<br>• Assess/evaluate/tradeoff alternative COTS-based solutions.<br>• Recommend an overall CBS strategy.<br>• Create a corresponding CBS plan, including contingency plans.<br>• Reassess and revise CBS strategy as necessary. | • Because of the presence of COTS products, the acquisition strategy/plan must be flexible enough to respond to changing circumstances. This requires, among other things, an understanding of how unprecedented your system is with respect to what the marketplace provides and with respect to what combinations have been successfully fielded for your application area in general and by your contractor. |

of control over many of these sources of contention and an inability to compensate by modifying COTS products.

**Cultural Transition Activity Set.** COTS-based systems represent a change for everyone in an organization, not just technical personnel. New roles and skills are required. Failing to pay attention to the cultural transition issues could result in a potentially insurmountable barrier to CBS success. The more a program already uses sound system engineering practices, the easier it will be to transition to a CBS approach.

**Information Sharing Activity Set.** Information sharing can help save others from repeating known mistakes. When the pace of change accelerates, as with COTS products and technologies, flexibility becomes a business imperative. A program does not have the time to dig itself out of problems that could be avoided.

## Future Directions

These results are preliminary. They require a great deal of application to validate and tune. We plan to use applicable activity sets with our customers, and we invite readers who choose to work with some or all of the activity sets, as described more fully in [8], to share their results with us. We expect to apply those results to evolve this preliminary framework.◆

## References


1. Boehm, Barry and Abts, Christopher, COTS Integration: Plug and Pray? *IEEE Computer*, Vol. 32, No.1, Jan. 1999, pp. 135-138.
2. Brownsword, Lisa; Carney, David; and Oberndorf, Tricia, The Opportunities and Complexities of Applying COTS Components, CROSSTALK, April 1998, pp. 4-6.
3. Garlan, D.; Allen, R.; and Ockerbloom, J., Architecture Mismatch: or Why It's Hard to Build Systems Out of Existing Parts, *Proceedings of the International Conference on Software Engineering*, Seattle, 1995, pp. 179-185.
4. Fox, Greg, Marcom, Steven, and Lantner, Karen. A Software Development Process for COTS-Based Information System Infrastructure, CROSSTALK, May 1998, pp. 20-25.
5. Reifer, Donald, Product Line Management: Best Acquisition Processes/Practices, July 30, 1999, Southern California Software Process Improvement Network (SPIN), University of California at Irvine, Irvine, Calif.
6. Sledge, Carol and Carney, David, *Case Study: Evaluating COTS Products for DoD Information Systems.* CBS monograph series. Pittsburgh, Pa.: Software Engineering Institute, Carnegie Mellon University (www.sei.cmu.edu/cbs/monographs.html) June 1998.
7. Hissam, Scott and Plakosh, Daniel, *COTS in the Real World: A Case Study in Risk Discovery and Repair*, SEI Technical Note CMU/SEI-99-TN-003, Carnegie Mellon University, June 1999.
8. Oberndorf, Patricia; Brownsword, Lisa; and Sledge, Carol, *An Activity Framework for COTS-Based Systems*, SEI Technical Report CMU/SEI-2000-TR-010, to be published.


## Notes

1. A COTS product is a product that is sold, leased, or licensed to the general public; offered by a vendor trying to profit from it; supported and evolved by the vendor, who retains the intellectual property rights; available in multiple, identical copies; and used without modification of the internals.
2. The term *system context* is used to ensure inclusion of requirements in the context of their end-user processes and other constraints such as cost and schedule—not just functional requirements.

## About the Authors


**Lisa Brownsword** is a senior member of the technical staff in the CBS Initiative at the SEI. Before joining the SEI, she was employed at Rational Software Corp., providing consulting to managers and technical practitioners in the use of and transition to software engineering practices, including CASE, architecture-centered development, product lines, and object technology.

Software Engineering Institute
4301 Wilson Blvd, Suite 902
Arlington, Va. 22203
Voice: 703.908.8203
Fax: 703.908.9317
E-mail: llb@sei.cmu.edu



**Patricia Oberndorf** is a senior member of the technical staff at the SEI. She is a part of the CBS Initiative and concentrates on the investigation of acquisition, management, and open system issues. Prior to coming to the SEI, she was with the Navy for more than 19 years, working on CASE environments.

Software Engineering Institute
4500 Fifth Ave.
Pittsburgh, Pa. 15213-3890
Voice: 412.268.6138
Fax: 412.268.5758
E-mail: po@sei.cmu.edu



**Carol A. Sledge, Ph.D.,** is a senior member of the technical staff at the SEI. She is a member of the CBS Initiative at the SEI and concentrates on open systems and acquisition and management issues of COTS-based systems. Sledge has 22 years of experience, primarily acquiring, developing, and supporting large, multi-platform product line systems.

Software Engineering Institute
4500 Fifth Ave.
Pittsburgh, Pa. 15213-3890
Voice: 412.268.7708
Fax: 412.268.5758
E-mail: cas@sei.cmu.edu


**UPDATE**

***The Guidelines for Successful Acquisition and Management of Software Intensive Systems (GSAM)* is available free on the CD-ROM distributed by the Software Technology Conference 2000 only to those people who attended the conference. Others may purchase the CD for $50 from Utah State University. For a copy, contact Vivian Johnson at (435)797-0424 or vivian@ext.usu.edu.**

**The GSAM is also available on the Software Technology Support Center's web site at www.stsc.hill.af.mil, and will be included in a future release of the Defense Acquisition Deskbook. No hard copies will be available.**

# Supporting Commercial Software

Lt. Col. Lionel D. Alford
*United States Air Force*

*Commercial and Nondevelopmental Items (CANDI) has become a byword for acquisition reform, but there are significant risks associated using CANDI products in military systems. These risks are especially acute for software. This paper explains how CANDI can negatively affect military acquisitions and gives ideas on how to plan and resolve CANDI-caused problems.*

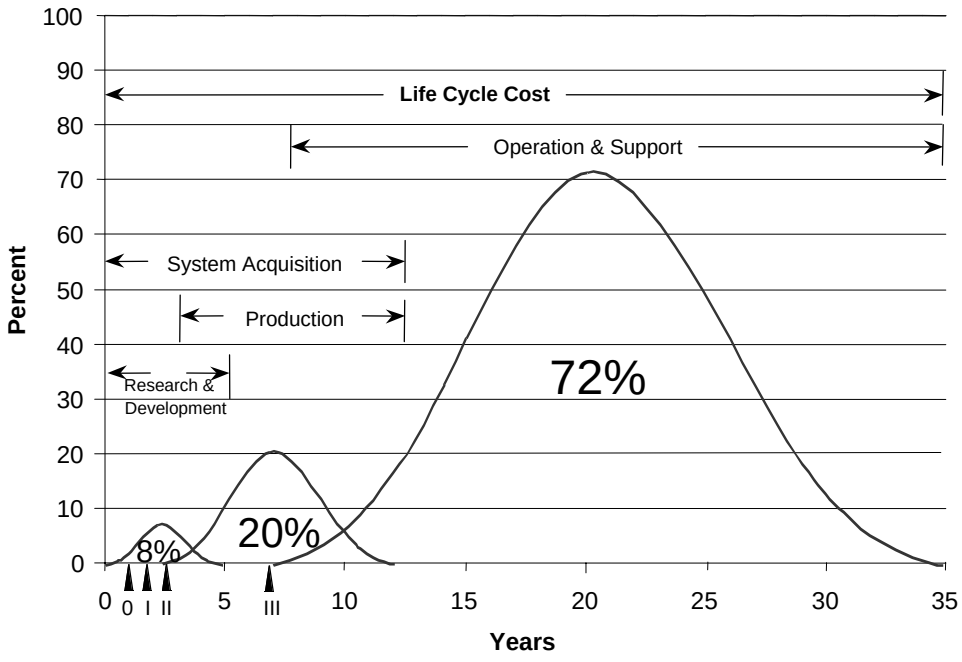
To take advantage of the fast pace of technological advances in industry, the Department of Defense (DoD) is acquiring commercial products and components, called CANDI, for use in military systems. CANDI provides the DoD with numerous potential benefits.

Primarily, commercial purchases allow military acquisition to incorporate new technology into military systems more quickly than typical developmental programs. CANDI also can reduce research and development costs. Even more importantly, the DoD has looked to commercial purchases to help reduce operations and support costs for military systems. Figure 1 shows why the DoD finds this highly desirable; the cost of operations and support is almost three-quarters the overall cost of a typical system. What could be the worst misfortune to befall software procured as CANDI—that the software changed and the original version was no longer available commercially? What if the commercial replacement would no longer work in the military system for which it was procured? The absolute worst misfortune that incorporates both of these problems is if the software were to suddenly become government-unique—no replacement was available commercially. Becoming government-unique would not entirely defeat the purpose of a commercial software acquisition, but it would significantly affect support, the longest tail and as shown in Figure 1, the greatest cost in the acquisition life cycle. This misfortune could never affect our commercial procurement, or could it? When you have finished this article, you will realize that not only can it affect your commercial procurement—if you are acquiring software, it probably already has. In any commercial acquisition, the acquirer needs to plan for this eventuality. This article will show you how to prepare for and give you ideas on how to constrain this problem.

An item is "government-unique" when the government is the only source of the item—this is the conceptual opposite of a commercial item. In terms of logistical support, an item is a discrete unit that can be individually acquired for the logistical support of a system. Software, in this definition, is an item while a system is the higher-level mission component the item is procured to support. For example, an aircraft and its support equipment are a system; a radio installed in the aircraft is an item, and the software that integrates the radio into the aircraft is an item. Whenever a manufacturer discontinues or makes a change to a commercial item, the item can become government unique. When the manufacturer changes the item, if the government does not acquire the variant or does not reflect the change in the systems incorporating the item and the systems' documentation, the original becomes government unique. After a manufacturer makes a change to an item, the government might be able to purchase and use the new variant without any negative effect to the system. In this case, although the original item is now government unique, the change did not affect the form, fit, interface, or mission characteristics of the device. Unfortunately, manufacturers' changes routinely affect form, fit, interface, and mission characteristics, and the effects of these commercial item changes for systems incorporating them are significant. The problems of changing form, fit, and interface should be obvious; if the variant item is to be installed and operate correctly, these characteristics cannot change. To accommodate form, fit, and interface changes, the acquirer must make modifications to the system. Modifications are costly and usually result in the original item becoming obsolete. Changes to mission characteristics do not necessarily result in system modifications, but if they affect the overall ability of the system to perform, they can cause significant problems. For example, if the new software version incorporates undocumented features or unnecessary compatibility, the entire system's security could be at risk.

Although software configuration changes can cause havoc in any program, the most devastating cause of government

Figure 1. *Typical Cost Distribution*[1]



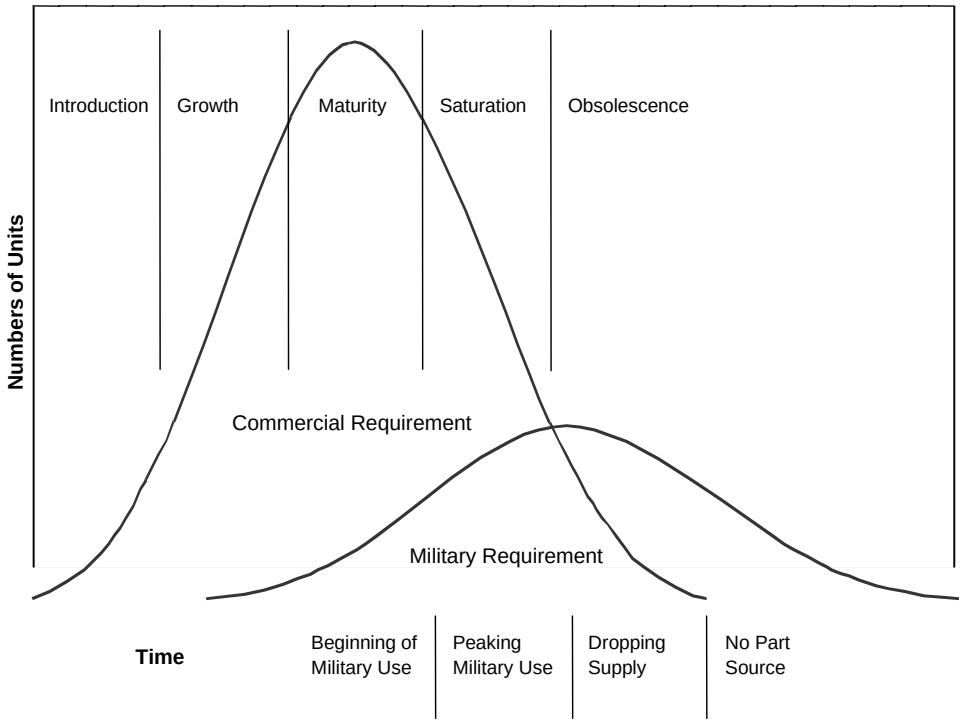



Figure 2. *COTS Obsolescence*[2]

uniqueness occurs when a manufacturer discontinues an item. Figure 2 shows that this is inevitable for a large number of commercial acquisitions. The life of a typical military acquisition exceeds 20 years, yet the life of a typical civil product, especially in electronics is much less. This example is for hardware, but the critical integration of most of our modern hardware systems is via software. Not only does the software have to change to accommodate changing items, but as hardware improves software must continue to increase in capability. Through experience with computer hardware, we know an "ancient" Z80-based computer is almost impossible to purchase, but now the IBM 1750 chipset, a 5 MHz Z80 generation processor, lives on in the Air Force's AP-102 computer. The critical difference with software is that although the IBM 1750 still powers the AP-102, the software that interfaces and operates the AP-102 has changed almost yearly since the first fielding of the AP-102.

The above concepts provide the definitive framework under which commercial software must be understood: without notice, the manufacturer is free to make changes to or discontinue the manufacture of the commercial item and its supporting software. As long as the manufacturer's item changes do not affect form, fit, interface, or mission characteristics the acquirer has no problem. The problem is that the acquirer has no control over these changes, and when changes do affect form, fit, interface, or mission characteristics, these changes become a significant problem for any commercial acquisition. This is especially true for aviation CANDI.

The effects of a manufacturer's changes to aviation CANDI can be boiled down to two specific difficulties—airworthiness and forced modifications. Airworthiness is the primary safety characteristic of any aircraft. It is the primary element proven in the testing of the aircraft. The FAA certifies the airworthiness of commercial items for aircraft, and these items must be certified in the system as well as individually. In addition, certification is an inherent governmental responsibility that cannot be delegated to a contractor [1]. Because of this, military system certification, except for FAA-certified aircraft, is accomplished wholly by the aircraft's configuration management (CM) authority. In the Air Force this authority is the Single Manager (SM). What this means for CANDI software is that a simple change of mission characteristics, including improved functionality, will always drive a recertification of the aircraft. This recertification can range from a paper review to full flight test. The rate of change in commercial items can be significant. This is especially true for aviation CANDI. Considering the rate of change of commercial items, frequent recertification is a daunting prospect for the CM authority. In addition, commercial item changes also can drive changes to the specifications and technical data of any system on which these items are installed, a daunting prospect.

Forced modifications are the other difficulty for aviation CANDI that also affects any system. A forced modification is a system's modification caused by the change of form, fit, interface, function, or mission characteristic of the item. When a change affects a mission characteristic, the acquirer must support the discontinued item or find a replacement. The later may force a modification. More common in aviation CANDI is a FAA-directed change to an item called an airworthiness directive (AD) [2]. Airworthiness directives are Federal Aviation Regulation-based orders that mandate a change to an aviation item or system. These directives are regulatory in nature and "no person may operate a product to which an airworthiness directive applies except in accordance with the requirements of that airworthiness directive [2]." The manufacturer has two choices in implementing the AD: discontinue the product or make the required change. The user of the item also has two choices: find a replacement product, if available, or make the changes required by the directive. When the change affects the form, fit, or interface of the item, an AD forces a modification to the system to accommodate the item. For FAA-certified aircraft, the FAA must also certify the system for flight. For government-certified aircraft, the CM authority must modify the system and certify airworthiness in order to comply with an AD. However, the government is under no obligation to change its commercial items to accommodate an AD. If the government does not change a commercial item to comply with an AD, the item becomes government unique. Because the government self-certifies, commonly, non-FAA certified government aircraft do not make AD directed changes. Further, because in many cases the government does not subscribe to technical changes from manufacturers, the CM authority may not be aware of ADs to a system's

components or software. This problem is exacerbated when the SM has established a depot for a commercial acquisition and is, in that case, supporting the component without knowledge of, or real commonality with, the original item. ADs are not an isolated or uncommon problem. Typically on well-established airvehicles, ADs normally occur more than once per year, and thousands of ADs may affect a single aircraft model.

All this boils down to the fact that, for aviation, a commercial item will become government unique in a very short period of time—from a few months to a year following the acquisition of the item. Government uniqueness means forced review, modification, support changes, and recertification when the change is recognized, or blissful ignorance and risk if the change is not recognized.

Change to aviation CANDI is not the only certification problem that confronts the SM or Designated Approval Authority (DAA). Communication and intelligence $C^2$ software and hardware systems require their own item and system certifications. The terms "networthiness" and "infoworthiness" are beginning to come into their own. These terms refer to the security of hardware and software systems connecting communication and intelligence infrastructures. Any changes to an item or software requires recertification of the system for security. For communication and intelligence $C^2$ software and hardware systems this is the DAA's responsibility. In all other cases, the CM authority must certify the security of the software.

## Commercial Support Strategies

What can be done to prevent these problems for software systems specifically and all systems generally? One solution has been hinted at, and this solution has been accomplished with varying degrees of success since the first acquisition of commercial items.

### Organic Support

This approach is the acknowledgment of an item's potential government uniqueness before the manufacturer makes any changes. In this strategy, the acquirer purchases spares and builds a government depot activity to support the item. This solution does take advantage of the original commercial item development, but the overall cost savings may not be significant because the longest tail—the support tail, is at least as long as any normal government item development. In fact, the support tail may be costlier because the government has not been involved in the item development. Many programs use this strategy; the C-130 improved auxiliary power unit program is one example.

### Lifetime Spares

Another similar solution is to purchase enough spares for the total life of the system and item. The AP-102 computer program used this strategy to ensure sufficient IBM 1750 chipsets to support the life of the system. Again, this is not an optimum solution because it usually increases the item's logistics tail. In this case, if the item's life expectancy is less than predicted or the item's life is extended, the government has no other recourse than to entirely replace the item or to develop a support capability. Further, although lifetime buys might seem to freeze software changes, as the example of the AP-102 computer shows this is rarely the case.

These two solutions, government organic support and lifetime spares buy, prevent forced modifications and subsequent airworthiness certification requirements, but as discussed above, they also can introduce risk. They also defeat two major potential advantages of CANDI—the ability to reduce the support tail and the ability to take advantage of future commercial developments in the item.

There are four other solutions that take full advantage of the possibilities of commercial acquisition, but they are each fraught with their own risk. Each of these four solutions is a variant of what is commonly known as Contractor Logistics Support (CLS). In this context CLS does not refer to basic maintenance support but rather to data and software support of modifications to support changes to commercial items.

### Purchase Technical Information

In the first alternative, the acquirer can purchase the manufacturer's servicing information support. This allows the CM authority to make decisions based on changes to the item. If the CM authority knows of a manufacturer's changes to an item, they can choose to acquire a replacement or modify the system as required to allow continued use of the variant item. The SM has three options:

- First, when an item changes, if it is decided to replace the item, the SM must acquire and certify the new item.
- Second, if the item is retained with changes, the SM must certify and possibly modify the system.
- Third, if it is decided to not make any changes to the item, the SM must set up government-unique support.

The advantages of retention or replacement (Options 1 and 2) are the continued commercial logistics tail and guaranteed item certification. The SM must still recertify the system. If the item is retained in its original configuration (Option 3), the decision to support a now government-unique item leads to a typical high-cost government logistics tail. To my knowledge, this pick and choose method of systems support has not been used intentionally; however, after a manufacturer has made unexpected changes to a commercial component, many programs have found themselves in this situation.

### Purchase Manufacturer Support

The second CLS alternative is that the acquirer can purchase manufacturer support for the item. The risks are similar to purchasing servicing information support; however, the manufacturer has potentially greater incentive to keep the item within form, fit, and interface configuration for the system. When changes in the system are required to support changes in the item, the manufacturer can aid the CM authority. This is a common method used to support CANDI.

### Purchase Manufacturer Modification Support

In the third alternative, the acquirer can purchase the full-integrated support of the manufacturer. This allows the manufacturer to make changes to the system along with changes to the item. The contractor may have some Total System Performance Responsibility (TSPR), but the CM authority must still recertify the system. The

AC-130U is using this method to manage CANDI in its new Integrated Weapon System Support Program. This is the most successful method used today to support commercial items and systems through CLS; however, it requires a continuing commitment to the manufacturer and to support funding.

### Purchase Full Manufacturer Support

Fourth, the acquirer can purchase full system support that would allow an integrator to automatically make the necessary changes to the system to accommodate any item changes. In this scenario, the contractor would have TSPR and a government agency other than the military (the FAA, for example) would certify the weapon system. This option is used now primary to support FAA-certified government aircraft. It could potentially be used to support any government aircraft or system incorporating commercial items. A problem with this method of support is that FAA certification of aviation systems may not fulfill military requirements. In addition, the DAA or SM must still certify system security.

## Conclusion

The message should be plain. Commercial acquisitions lead the acquirer down two support paths: the government unique high-cost logistics trail and the commercial manufacturer support trail. Both involve risk and guarantee future costs for any system incorporating commercial items. The potential of commercial acquisitions is embodied in a lower cost development, initial acquisition, and support costs, but that potential must be balanced with the knowledge that commercial acquisitions will force modifications and recertifications or lead to a typical government unique logistics tail.

CANDI software is a viable method of military acquisition, but it is not a simple solution. It requires careful planning and forethought that must be incorporated into any program contemplating a commercial acquisition.◆

## References

1. Office of Federal Procurement Policy. (Sept 92). OMB Policy Letter 92-1, Inherently Governmental Functions. OFPP Pamphlet No. 6 (Revised), Fourth Edition, December 1992. Washington, DC: Government Printing Office.
2. Federal Aviation Administration. (Feb 1996). Part 39—Airworthiness Directives, Federal Aviation Regulations. Washington, DC: Government Printing Office.

See www.safetydata.com for an example of Airworthiness Directives.

## Notes

1. Source for figure is Defense Systems Management College. (December 1997). T5-610 Acquisition Logistics and Systems Engineering, Technical Perspective: Logistics Management.
2. Source for figure is Obsolescence Crisis. Joint Stars: Joint Surveillance Target Attack Radar Briefing.

## About the Author

**Lt Col Lionel D. Alford, Jr.** is the Aeronautical Test Policy Manager for Air Force Materiel Command, Wright-Patterson Air Force Base, Ohio. He is an Air Force experimental test pilot with over 3,600 hours in more than 40 different kinds of aircraft and is a member of the Society of Experimental Test Pilots. He has served as the Chief, Special Operations Forces Test and Evaluation Division at Wright-Patterson, Chief, Testing Commercial Aircraft for Military Acquisition Office at Edwards Air Force Base, Calif., holds an Airline Transport Pilot license, and was the chief test pilot for a number of Air Force acquisitions. He is a graduate of Defense Systems Management College Advanced Program Management Course. He has a master's degree in mechanical engineering from Boston University and a bachelor's degree in chemistry from Pacific Lutheran University.

Lt. Col. Lionel D. Alford Jr.,
USAF, HQ AFMC/DOP
Bldg. 262, Room S143, 4375 Chidlaw Road
Wright-Patterson AFB, Ohio 45433-5006
Voice: 937-257-8496
Fax: 937-656-1246
E-mail: Pilotlion@aol.com,
Lionel.alford@wpafb.af.mil

# Evaluating COTS/GOTS Software: Functional Test Criteria

Dr. William H. Dashiell
*DoD National Imagery and Mapping Agency*

Phil Brashear
*EDS Conformance Testing Center*

*As government agencies move toward commercial off-the-shelf (COTS) and goverment off-the-shelf (GOTS) software, they realize that vendor descriptions don't always give sufficient information about the products' capabilities. Program Managers (PMs) need to consider not only functionality of software for satisfying user needs, but also compatibility and performance in the program's environment. Unfortunately, PMs responsible for acquiring software, although quite good at developing screening requirements, are often not experienced in writing testable requirements. This paper is addressed to people in that position.*

Planned and scheduled software testing—whether unit testing, integration testing, conformance testing, etc.—is a basic element of risk reduction in software acquisition. When software testing is viewed as unsuccessful, it often turns out that fundamental supporting concepts have not been fulfilled. Contributing factors to unsuccessful testing include:

1. Mistaken concepts of software testing definitions, objectives, or applications
2. Disagreements with customers or users regarding methodology, standards, or interpretation of results
3. Budget reductions or insufficient budget to support the required testing effort
4. Conflicts between testing schedule and product acquisition schedule. Too often the product acquisition schedule wins this conflict.

More and more agencies are discovering the need for carefully inspecting functional requirements and formal testing products for conformance to these requirements.

In the following scenario, we meet Pete Anthony, a PM assigned to acquiring a COTS or GOTS Financial Accounting Package. Some problems have arisen, and it has been recommended that he consult with Tom Edward, a contractor Testing Expert (TE) who had completed other software testing for the group. The scenario begins with the arrival of the TE at the PM's office.

## Introduction to the Problem

PM: Good morning, Tom. Thanks for coming by.

TE: Glad to be here, Pete. What can I do for you?

PM: You know that my Personnel Management System acquisition went well, so I was tasked with acquiring a Financial Accounting Package. That did not go so well.

TE: What do you mean?

PM: Initially, I talked with prospective users of the package and formed a Product Selection Team (PST) to develop the selection requirement, selection process, and selection schedule. We interviewed users and managers, reviewed literature from various vendors, and wrote our selection requirements. Based on these requirements, the product literature, and user comments we invited five vendors to give demonstrations. The demos indicated that several of the products met our selection requirements. One product appeared to be outstanding in terms of functionality and cost.

Linda, one of the PST members, had recommended that we include an in-house testing process to reduce our risk of wrongly selecting a product. We followed her recommendation and included testing in the selection process and schedule. After the demos, another member raised a serious question. Everything we had done so far had taken a little longer than planned. Would we be able to have the new system running by the planned operational date? To put our problem another way, which was the greater risk, failing to meet the delivery schedule or selecting an inadequate product?

This started us thinking about the testing process again. We discussed it at some length, and finally concluded that the cost/benefit ratio did not justify testing. The vendors' literature and the demos showed that the remaining candidates would meet our needs, so there was no significant risk.

TE: Uh-oh.

PM: Exactly. That was a mistake. What we thought was a small risk, was not.

I directed the Contracting Officer to finalize the contract with the selected vendor, and the vendor arranged for installation. I notified the potential users of the installation, training, and data migration schedule, and told them that the COTS Financial Accounting Package would be operational in 60 days.

The vendor came in to install the software. After five days without success, the vendor pointed out that the contract called for payment of $150 per person per day to complete the installation. I was upset and demanded to know what was wrong. After all, the demo was perfect! The vendor representative explained that the demo was performed on a different hardware and operating system environment. The product was not really ready to operate on our LAN, which has servers running Linux and clients running Windows NT.

## Reducing Risk with Software Testing

TE: What did you do?

PM: I had to cut my losses. I directed the Contracting Officer to cancel the contract, notified the team and my supervisors of the cancellation and the reasons for it. My supervisors were pleased that I had stopped the process before any more damage was done, and asked me to pick up the pieces and restart. The first thing I did was call Linda, who recommended testing in the first place to ask for advice. We discussed how testing should have caught the problems and that neither of us knew enough about testing to do it right. She suggested I call you to see what pointers you could give me.

TE: Sounds like what we need to talk about is conformance testing against your requirements.

PM: I am not sure that I know exactly what you mean by conformance testing. Could you be more specific?

TE: Conformance testing is simply formal testing of a product against a set of conditions to which it must conform. For example, my organization tests compilers for programming languages, display software for Computer Graphics Meta-file (CGM), and other products. The conformance requirements are provided by international standards, and our testing is highly formalized, using extensive suites of tests that have been validated by experts in the various fields.

PM: Why would anyone need that level of formality? Is it not enough that the supplier implemented ISO standard C++ in his compiler or that his CGM interpreter is widely used in industry? If more assurance is required, can we just verify that the compiler processes our own programs correctly, or

that the CGM interpreter properly displays the pictures that we care about?

TE: In this information age, we have come to depend on software to create, process, transmit and store critical information. Our missions live and die on the correctness and reliability of such software. Just consider the rapid growth of electronic commerce in both the government and commercial worlds. For instance, the Department of Defense (DoD) has hundreds of sites all over the world that are processing contract data. They rely on the correct behavior of the software to keep that data correct, secure, and synchronized. Airframe manufacturers depend on CGM interpreters to display design diagrams in hard copy and on screens; if those diagrams are not correctly rendered, costly errors can result. Weapons systems depend on software for targeting, for vehicle control, and for stores management. If a compiler does not translate the programmer's intent correctly, warfighter and civilian lives can be threatened.

PM: OK, I can understand the importance for high-risk situations, but is conformance testing really needed for my Financial Accounting Package? When I buy a television set, I do not subject it to all sorts of tests.

TE: That is because first of all, you know that if it does not work correctly, you can get satisfaction from your warranty. You might miss a football game or episode of JAG, but no long-term damage is done. However, can you afford the loss or corruption of your Financial Accounting Package data? Can you afford the down time when payroll checks are due? If the information is important enough to be managed, then it is too important to risk. The consequences of nonconformance for your Financial Accounting Package may not be life-threatening, but the inconvenience could be pretty severe. You apparently have already experienced the problem when you could not install the selected software package.

PM: That is true. I agree that conformance testing is what I need. We already have our selection requirements, so all we need to do is develop tests.

## Reducing Risk—Defining Test Requirements

TE: What are some of your requirements?

PM: Here are a few examples: One of our data requirements is that the product must provide a flexible and common tracking numbering system. Under security requirements, it must support multiple levels of administrative access to the data. Finally, one of our interface requirements is that the product must be able to export data in a format readable by a spreadsheet program. What do you think?

TE: To be polite about it, I think they need work.

PM: Why? Aren't they sufficient to separate sheep from goats?

TE: Exactly, but I could not build a test for any of them. They are screening requirements, not testable requirements. They are at a pretty high level of abstraction. The granularity of your requirements is too large to allow me to detect the differences we need to detect if we are to see the risks for each choice we may take. You will need to develop testable requirements based on your screening requirements.

PM: Screening requirements, testable requirements. What is the difference?

TE: Screening requirements help to identify software that offers functionality similar to what you need; that is, separate the sheep from the goats. You have that. What you need now is assurance that this functionality performs as advertised and that your needs are really met. That requires more detail. I would expect each screening requirement to lead to several testable requirements, each addressing a specific facet of the screening requirement.

Look at these one at a time. Your first requirement calls for a "flexible and common" numbering system. The first thing that I notice is ambiguity. Do you mean that the numbering system must be common to multiple features, or that the product must allow the use of a numbering system that is commonly used in your organization's tracking? If the latter is what you want, then what are the candidates? How can we define *flexibility*? What kind of flexibility do you need, and to what degree? Finally, what kinds of entities need to be numbered? The test team has to know exactly what you want in order to produce satisfactory tests.

PM: I think I see what you want. We need to specify this requirement more carefully. I could start with something like this: The product must provide a system of numbering data items that allows association of expenses and revenues with projects, bids and proposals, training efforts, and other activities. The numbering system must be multilevel and must be consistent across all uses; for example, the first portion of the identifier might be used to identify the category (service project, bid/proposal, training, etc.) while the second might be used to distinguish expenses and revenues. The product must allow the agency to define the number of levels and the meaning and format of each.

TE: That is a good start. I still cannot write tests based on this requirement, but you have removed a good deal of the ambiguity. Let's use your second requirement to illustrate what is still needed.

PM: That is the security requirement. We want multiple levels of data access.

TE: Yes, but how many levels do you want? How do you want the access levels to be determined? The problem is that the requirement is at a high level of abstraction. The granularity is too large to allow us to detect the differences that indicate the risks for each choice we may take.

PM: What do we need to do to this requirement?

TE: We need to apply an iterative process, breaking it down into more specific requirements, then breaking each of those down again, repeating until we reach individually testable requirements. The first breakdown might include:

- Distinguishing among no access, read-only access, and write access.
- Determination of access rights by user ID and password.
- Determining access rights by project, determining by expense items vs. revenue items, or by forecast vs. actual.
- Giving the system security manager the ability to change access rights by individuals or groups of users.

PM: It is a matter of specificity vs. generality?

TE: Correct. Eventually, we must have requirements such as, "The product shall permit the system security manager to change a user's level of access to the revenue data for an existing project, from any level to any level, while the user is viewing such data."

PM: Is that possible for most database applications?

TE: I am not proposing this as an actual requirement, but giving it as an example of a testable requirement. You have to decide whether it is a requirement.

PM: There is clearly a lot of work to do; what about the interface requirement?

TE: That is not really a conformance requirement, but more of an interoperability requirement. You might require that the product is able to export selected data in a particular format; you cannot require that another product, such as a spreadsheet program, accept that format. The way your requirement is stated, you are trying to do both.

## Interoperability Testing vs. Conformance Testing

PM: Isn't interoperability testing part of conformance testing?

TE: It can be, depending on how you view the system. If you are merely interested in the functional requirements of the Financial Accounting Package, the answer is no. Interoperability testing focuses on the interfaces among different products, so the functionality of other products comes into play. If you view a collection of products as a complete system, and your functional requirements are stated for the entire system, the components' interoperability is part of conformance testing. Otherwise, I think of interoperability testing as a separate activity.

PM: Is any part of the interface requirement related to conformance testing?

TE: I think so. For example, most spreadsheet programs have the capability of reading textual tables of data where the column entries are separated by commas or by tabs. You can require that your package is able to output comma-delimited text files representing tables of data (although you would need to state it with more specificity). We could develop a conformance test for such a requirement. It is the spreadsheet program's ability to accept the comma-delimited format that we cannot handle as a conformance issue.

PM: I think I see where to go with the interoperability issue. Now let's focus on conformance testing of the single product. You have implied that it might need to be pretty extensive. How do I find the time and money?

## Risk of Not Testing

TE: The question is whether you can risk skipping conformance testing. You are looking at a major investment in this Financial Accounting Package. You have seen the consequences of trying to save money and time up front at the risk of spending more money and more time later on. Never mind the other consequences of corrupted or lost data. It is like a design-and-build activity; investment of resources early in the process mitigates against risks later, when costs and consequences are more significant. I strongly recommend including requirements development and testing in your budget and schedule.

PM: It is clear to me that this conformance testing is beyond my level of expertise. Where do I go to get help?

## Software Test Team

TE: That depends on your organization. There might be a suitable testing group within your agency. On the other hand, you might want to bring in a non-government group. You should expect better results if the test team has no interest in the outcome. You want a test team that has no legal or financial ties to the prospective suppliers or to your acquisition effort. You are trying to reduce risk, and that means you want the most objective viewpoint possible.

PM: What services should I expect from a testing group?

TE: First, it should work with you to derive functional requirements from your screening requirements. At each stage of this iterative process, it should work to ensure that its understanding of the requirements is the same as yours.

Second, it should decompose the functional requirements into individual test cases, with pass/fail criteria for each case.

Third, it should perform the actual testing, using the deployment system or an exact replica of it. Fourth, it should produce the required deliverable products

PM: Your first expectation relieves my mind. I was worrying about this process of deriving testable requirements, but you are saying that is one of the test team's jobs, working with us. That is the source of expertise that I am missing.

TE: Yes, this is the point where cooperation is the most important. The test team has the expertise to derive testable requirements, but only your organization can say, "Yes, that is what we really want."

PM: What are the deliverable products that I should ask for?

## Software Testing Deliverables

TE: First and foremost, a complete test plan. This document would probably be delivered several times, and would eventually include the original requirements, the functional requirements with traceability to the screening requirements, the test cases, procedures for executing the test cases, and a complete schedule for the testing effort.

Second, a test log should be provided documenting the steps as actually performed. Each individual tester should keep a complete diary of his or her testing activities. The test log should be constructed by integrating these individual testing diaries. The test report should have an executive summary providing enough information to allow informed decision making.

PM: Have we covered everything? Is there anything else I need to consider?

TE: There is one thing you should be prepared to face. It is quite possible that none of the candidate products will pass every test case. You should consider some prioritization or some plan for providing missing functionality if no product has absolutely everything that you need. That is a knotty problem, and its solution is not exactly in the center of my area of expertise.

PM: Tom, I appreciate your advice, and I think I have no choice but to accept it. If we decide to look to an outside organization for testing services, I will certainly include your company on the list. Thanks again for coming by.

TE: Thanks, Pete. See you later.

## Summary

**Why do I need conformance testing?** In the world of electronic commerce, we are totally reliant upon software to create, process, transmit, and store critical information. We must be able to trust this software to correctly and reliably provide the required functionality.

**Why are my preliminary screening requirements unsatisfactory?** Screening requirements help to identify software that offers functionality similar to what is required, but we need assurance that the offered functionality performs as designed or as advertised to satisfy our needs. Screening requirements are given at a high level of abstraction, such as "The product shall be Y2K compliant." Testable requirements are derived from screening requirements, often with a testing team's help, and address specific items (e.g. "The product must accept Feb. 29, 2000 as a valid date and must reject with a warning message Feb. 29, 1900.")

**Is interoperability testing part of conformance testing?** Whereas conformance testing attempts to determine whether a product meets its functional requirements, interoperability testing focuses on the interfaces among different products. If a collection of products is viewed as a complete system, and functional requirements are stated for the entire system, interoperability of the components is part of conformance testing. Otherwise, we view interoperability testing as a separate, related, activity.

**How do I find the time and money for conformance testing?** In the case of a major investment of COTS software, you must consider the consequences of not doing conformance testing.

If your agency discovers that an already purchased product does not meet the agency's needs, money and time have been wasted. Recovery from this situation is likely to cost far more money and take far more time than would have been consumed by testing. Requirements development and testing must be planned as part of the acquisition cycle.

**How do I acquire conformance testing services?** That depends on your organization. The conformance testing team should be a third party, not associated with the acquisition group or the vendor. Conformance testing could be performed by a team outside the organization or from a separate division of the acquiring agency. It is important that the testing team have no financial, legal, or other dependence upon the acquisition team.

**What services should I expect from my testing organization?** It should help derive functional requirements from the screening requirements, ensure that its understanding of these requirements is the same as that of the acquisition team, decompose functional requirements into individual test cases, identify pass-fail criteria for each test case, perform the testing, and produce the required deliverable products.

**What deliverable products should I expect from testing?** The test team should provide a complete test plan, including requirements, test cases, test products, and schedule; a test log documenting the steps performed; a test report showing the individual pass/fail reports; and an executive summary sufficient for the acquisition team to make a decison.◆

## About the Authors

**William H. Dashiell** is a computer scientist at the DoD National Imagery and Mapping Agency. He has worked on the development of software testing by statistical methods using bionomial models, coverage designs, mutation testing, and usage models. He has contributed to the development of conformance and testing protocols for federal, national, and international information technology standards. He has a bachelor's degree in business administration and in education, a master's degree in education technology, and a doctorate in mathematics education from the University of Maryland. He also has a master's degree in computer science from Hood College in Maryland.

William H. Dashiell, Ph.D.
U.S. Government; NIMA
Voice: 703-874-8219
Fax: 703-874-8841
Adwd@netkconnect.net

**Phil Brashear** is a senior systems engineer at EDS in Dayton, Ohio, where he leads the EDS Conformance Testing Center and Performance Software Quality Assurance on various DoD projects. He directed compiler testing efforts and test suite maintenance at CTA Inc. from 1989-97. He performed compiler validations and directed test suite enhancements at SofTech Inc. from 1986-89. Prior to that, he was a member of the mathematics and computer science faculties at Eastern Kentucky University. He has a bachelor's degree in mathematics education from the University of Kentucky, a master's degree in mathematics from Northwestern University, and completed course work and exams for a doctorate in mathematics at the University of Georgia.

EDS Conformance Testing Center
P.O. Box 24593
Dayton, Ohio 45424-0593
Voice: 937-237-4510
Fax: 937-237-4660
Internet: www.eds-conform.com

# Coming Events

**September 18-19**
*The Internet Challenge—The Utility Response to a .Com World* www.tdworld.com/marketing/interchall.htm

**September 26-28**
*2nd Computer Security & Information Assurance Conference*
www.certconf.org

**October 15-19**
*Object Oriented Programming Systems Languages and Applications Conference (OOPSLA 2000)*
www.acm.org/events

**October 23-25**
*Symposium on Operating Systems Design and Implementation*
www.usenix.org/events/osdi2000

**October 30-31**
*3rd International Conference on Practical Aspects of Knowledge Management (PAKM 2000)*
www.do.isst.fhg.de/workflow/events/index_e.html

PAKM2000

**November 10**
*Infomation Outlook 2000 (Australian Computer Society)*
www.acs.org.au/act/events/io2000/index.html

**November 16-17**
*ACM Conference on Universal Usability*
www.acm.org/sigchi/cuu

**December 4-7**
*International Conference on Power System Technology*
www.ee.uwa.edu.au/~aips/powercon

**December 11-13**
*Global Development Network Conference*
www.gdnet.org

**January 18-19**
*Measurement Science Conference*
www.msc-conf.com/findex.html#cfp2001.html

**January 25-27**
*21st Annual National CSIE Conference*
www.ryerson.ca/~csie/2001/engindex.html

**January 30-February 2**
*CIEC 2001 Odyssey: Industry & Education Engineering*
www.asee.org/conferences/html/ciec2001.htm

**February 7-9**
*Network and Distributed System Security Symposium*
www.isoc.org/ndss01/call-for-papers.html

**April 29-May 3**
STC *Software Technology Conference*
www.stc-online.org

# Implementing COTS Open Systems Technology on AWACS

By Lt. Col. Michael K. J. Milligan
*U.S. Air Force*

*The U.S. Airborne Warning and Control System (AWACS) E-3 weapons system required a modernization program for its aging mission computing system. Due to the significant technological and cost advantages of using commercially available hardware and software, a distributed, object-oriented, open systems, commercial off-the-shelf (COTS) approach was taken. This article presents lessons learned from the development and preproduction of the U.S. AWACS Step 1 Mission Computing Upgrade Program.*

The purpose of this paper is to share experiences the AWACS development team encountered during the integration of COTS technology into the legacy E-3 weapon system. Sharing these lessons with similar programs may be helpful in avoiding some of the problems the AWACS team experienced.

The mission computing upgrade program was initially conceived in the Fall of 1995 as a result of the System Program Office's and Air Combat Command's desire to fix critical maintainability shortfalls and at the same time, get three critical operational capabilities aboard AWACS. These capabilities were required to enhance operational situational awareness, and include a more accurate tracker (fusing radar and IFF data), improved symbol definition, and more detailed (and useful) map displays.

Due to the high operations tempo and funding constraints, the acquisition approach used in the mission computing modernization effort injects technology in two fundamental steps, U.S. Step 1 and U.S. Step 2. Step 1 injects fundamental open systems technology, migrating the mission computing system from a vendor unique or closed legacy system to an open architecture and provides a path for future migration and growth. Step 2 completes the modernization effort. An open system implies that the system interfaces are public domain, so the selection and integration of components should be analogous to the concept of plug-and-play. Open systems provide cost savings by allowing a number of vendors to compete for the various hardware and software components in the broader commercial market. The AWACS architecture will no longer be tied to a specific vendor selling unique components built to proprietary or closed interface standards. By opening the architecture, future upgrades and new mission capabilities may be integrated with minimal integration and testing requirements.

The computer modernization development program was a joint effort among the AWACS System Program Office, Hanscom AFB; Air Combat Command/ 552 ACW and Air Logistics Center, Okla.; MITRE Corp., Lockheed-Martin Federal Systems, Boeing Space and Defense Systems, and GEC-Marconi Hazeltine. Many COTS vendors also participated.

In 1999, after approximately three years of development, the U.S. Step 1 production program was cancelled in favor of a larger, more comprehensive upgrade program. This program would continue developing the same technologies and COTS strategies as Step 1 while expanding the overall effort. There are many useful lessons learned during development and testing of the U.S. Step 1 program that can be applied to future modernization programs within the defense community.

## COTS Considerations

With the introduction of COTS into the E-3, several aspects of traditional military weapon system design were modified or eliminated. Key areas include physical and environmental characteristics of the various COTS hardware components.

COTS hardware used in the U.S. Step 1 architecture is not specifically designed to operate in an airborne environment. In order to take full advantage of COTS, the design team needed to determine if certain components could be used. For example, the temperature range specified in the original AWACS system specification required that all electronics operate within the operating range of -55°C to +85°C (-67°F to +185°F). After flying the E-3 more than 20 years, Air Combat Command determined that such a wide operating temperature range was not necessary in most cases.

The requirement was modified to specify use in the 0°C to +50°C range (typical for most COTS electronics) and included changes to some existing operational procedures. This modification to environmental requirements provided the opportunity for use of an increased number of hardware components from various vendors. COTS hardware used in the U.S. Step 1 architecture includes single board computer cards, graphics accelerator cards, power supplies, cabinets, VERSA Module Eurocard (VME) enclosures, network interface cards, network switches, fiber optic cables, SCSI disk drives, 1553 I/O cards, and solid state memory devices. In addition, several COTS software components are used, including a real-time UNIX compliant operating system, map generation software, compilers, graphical user interface generators (X-Windows, for example), debuggers, and network interface software drivers. In addition to the many COTS components, some custom hardware and software was required to interface the U.S. Step 1 architecture to the remaining legacy system.

## Legacy Software

In terms of life-cycle costs, software upgrades and maintenance are the most expensive component of the overall mission computing architecture. The current mission software consists of a single computer program called the Airborne Operational Computer Program (AOCP), which consists of approximately 370,000 lines of code written in Jovial and assembly language. The AOCP is responsible for all functions, including basic operating system services, timing and scheduling, and all applications, including weapons control, surveillance, display control, communications, internal simulation and system maintenance. Since the AOCP is based on a complex cyclic executive, any changes or upgrades made to the AOCP requires exhaustive functional and temporal testing to ensure

new functions operate correctly—logically and within specific timing constraints.

## Mission Computing Hardware

The U.S. Step 1 Mission Computing Hardware Architecture is shown in Figure 1. The shaded areas indicate those components being added or modified. As illustrated, there will be a mix of new and legacy hardware. The new architecture is designed as a client-server network, distributing functions among a number of processing nodes. Each node consists of a processor, Local Area Network (LAN) Interface Cards (NIC), and other specialized cards. All are based on the industry standard VME bus design. The processor family of choice is PowerPC due to its performance, support of real-time operating systems, and large market share for embedded real-time applications. The LAN protocol chosen is switched Fibre Channel, based on bandwidth and real-time support requirements.

One of the key differences between the legacy system and the new U.S. Step 1 design is the use of a client/server architecture. Client/server is a relationship between processes running on separate machines (processors). The server process is a provider of services, while the client is a consumer of services. In essence, client/ server provides a clean separation of functions based on the idea of service [1]. The overall goal of this new architecture is to ensure the ability to provide inexpensive, timely upgrades and/or modifications to any hardware or software component without directly affecting the overall architecture.

## U.S. Step 1 Software

Due to the many limitations and costs associated with development, maintenance, and testing legacy software, a modern software design consisting of a three-layered, object-oriented architecture was chosen for the U.S Step 1 Program. This new architecture is designed to allow migration of new software applications to a completely object-oriented design. The Distributed Software Infrastructure (DSI) is designed to isolate the application software from the operating system and allow encapsulation of all applications. This eliminates any application program dependencies (data or timing) on the operating system or specific hardware, and allows software to be developed and tested independently. This middleware is built according to open industry standards, ensuring that all present and future applications will communicate directly without any special, unique code changes. These open interface standards are called Application Programming Interfaces (APIs) and are based on the Object Database Management Group's (ODMG)'s Common Object Request Broker Architecture (CORBA) standards. By adhering to the ODMG's defined APIs, code developers can ensure their applications will interface correctly in any CORBA compliant environment.

The architecture supports this approach by being implemented as a collection of objects, and providing a framework in which objects can be shared among the distributed components of the AWACS computer system. For example, the display system application interacts with the tracker application by invoking well-defined methods on the tracker's interface. This hides the details of the complex tracking subsystem to the rest of the system. Most importantly, these interfaces are defined by an Interface Definition Language (IDL) that gets precompiled, enforcing the interface definition and allowing large amounts of automatic code generation. By communicating via these well-defined interfaces, all dependencies of the display on the tracker (and vice versa) are eliminated [2].

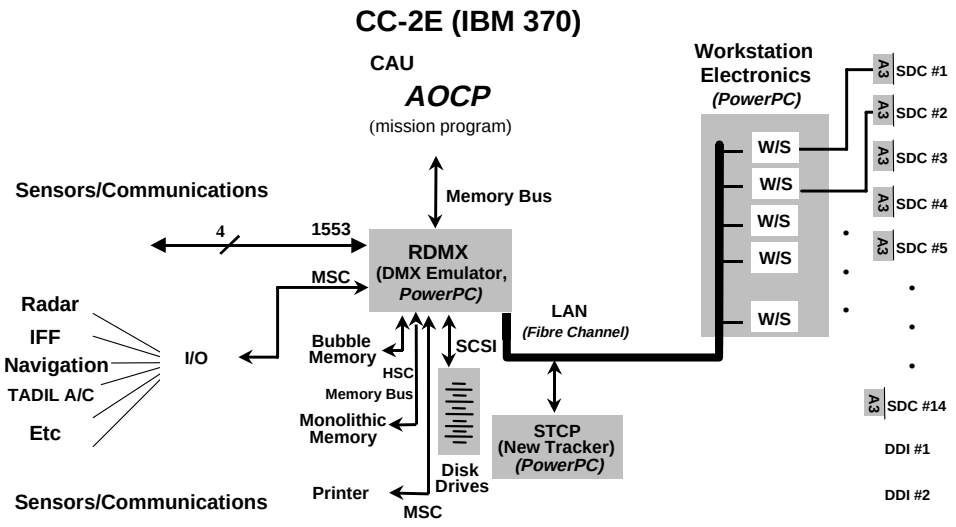
Figure 2 illustrates the software architecture. It is divided into three distinct layers: a real-time UNIX POSIX compliant commercial operating system,[1] software middleware (the DSI—information manager, encapsulated scheduler, and real-time database), and encapsulated applications layer. This design isolates all applications from the underlying operating system and any hardware dependencies, thereby ensuring platform independence for all applications. The DSI performs three vital functions:

1. It acts as communication pipeline, transferring objects between various component applications and database.
2. It schedules processes according to *a priori* priorities and pre-empts any lower priority processes if necessary.
3. It provides a real-time database.

A modified AOCP, with reduced functionality) continues to execute on the CC-2E (IBM 370) computer.

This software architecture is based on the philosophy of incorporating real-time design attributes into the architecture at all levels. The attributes of a real-time system may be characterized by the predictable response times; priority-based scheduling, and pre-emptive control. These three “P's” of real-time design allow the developer a great deal of control and flexibility, critical in designing today's complex real-time systems. Predictable response times under all load conditions ensure that applications respond to external events in a predictable fashion, regardless of what other tasks the system is handling. It requires consistent and prompt priority-based scheduling of time-critical tasks and low system overhead [3]. Pre-emptive control ensures deadlines are met by allowing lower priority process-

Figure 1. U.S. Step 1 Hardware Architecture (Functional Diagram)



es or threads to be pre-empted by high priority ones. The operating system, including the kernel, must also be able to be pre-empted. All three "Ps" are a function of the underlying operating system chosen to support the real-time applications [4].

## Lessons Learned

There were many technical and programmatic lessons learned that can be derived from the U.S. Step 1 development program.

***User buy-in to use of COTS is crucial.***

While recognizing the implementation of COTS-based systems as an enabler to modernization, AWACS operational users are extremely cautious of COTS in their day-to-day operations because the hardware was not designed to harsh environments. To ensure program success, it was critical to have their full support of the program and participation on a regular basis to ensure operational requirements were understood and met. There were many opportunities during program development for the users to partner with the developers to devise a solution, and their commitment to a COTS-based solution was critical. Educating the user and support community also was important. Since the operational user is not normally in the business of developing technology, there were many opportunities for misunderstanding, especially in the area of acquisition reform. This bold DoD initiative blends COTS-based solutions with streamlined management to produce superior products within tight fiscal and schedule constraints. Since the user and support depot were not necessarily current on this acquisition philosophy, conflicts often arose over development practices and perceived shortcuts. Some caution is also advised in the requirements area, since heavy user involvement at all stages of the program provides the opportunity for some requirements growth. This can lead to attempts to specialize the COTS away from the pure COTS baseline. Critical requirements must be solid.

***Establish a baseline with COTS.***

Since the COTS market is fast moving and ever changing, it was difficult to establish baseline architecture with COTS components if the program development schedule was longer than 18 months.[2]

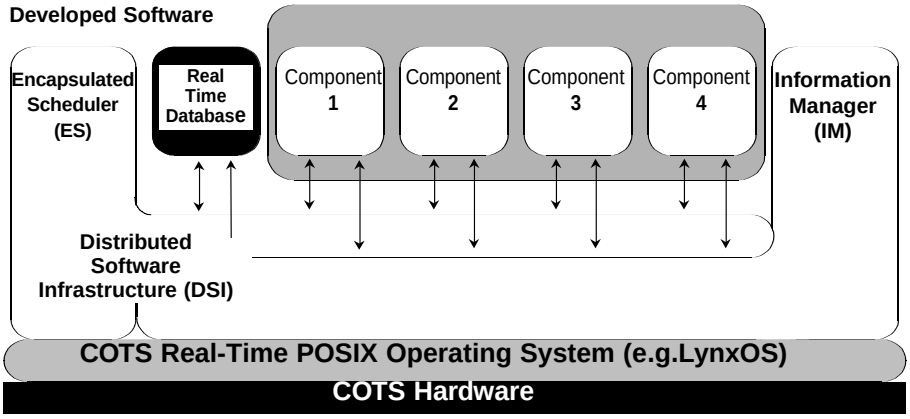



Figure 2. *U.S. Step 1 Software Architecture*

New products are introduced every six to 18 months, and often components chosen for the AWACS upgrades were phased out or no longer supported. As a result, it is critical to team with COTS vendors to ensure that the components chosen will last as least as long as the program's development and preproduction phases. This problem was especially apparent in the software area, where frequent new releases aimed at solving a select set of problems often produced new ones. Also, there was minimal documentation available from one software release to the next, so application developers had little insight into changes until anomalies were discovered.

***Transitioning contractors is not simple.***

Teaming with contractors with vast AWACS experience is vital to the success of any E-3 upgrade program. They have made a long-term commitment of building a specialized team of engineers, software specialists, and managers to support the unique requirements of a custom hardware and software system based on proprietary designs and technologies. However, object-oriented design techniques and implementation of COTS-based technology demands a change in design philosophy, subject matter expertise, and in many cases management structure. It was difficult for some contractors to effectively change and adapt in a timely manner. This was especially apparent in the software development area, where object-oriented software engineering techniques were not always well understood or implemented. This was primarily due to the nonavailability of modern software engineering and management skills within the company. Some contractors found it very difficult to hire new software engineers who possessed the necessary skills, due to the general market shortage of software professionals. The net effect was that software development and integration took longer than anticipated, which in turn severely impacted the overall development schedule.

***Be on the leading edge, but not the leader.***

More often than not, modern weapon system development is associated with leading edge technology. In the COTS-oriented world, while it is desirable to take advantage of the latest products, it is often painful to be one of the first users of a particular technology. You become the *de facto* beta tester of a product. Oftentimes, even if you are not the first user, you may become the first to stress the product's capabilities, sometimes discovering subtle deficiencies. This translates to additional time and money to troubleshoot and integrate. It is best to use COTS products that have some level of demonstrated maturity.

***Carefully investigate tool availability.***

The U.S. Step 1 program is based around the use of three programming languages: C, C++, and Ada. While implementing these languages for different parts of the design is not a problem, availability of advanced development tools was a challenge. For instance, since C and C++ are widely used, there are numerous software tools readily available to the software developer. However, while Ada was widely adopted by primarily government contractors over the past 15 years, other commercial developers did not embrace the use of Ada. As a result, development of advanced software tools for Ada lagged behind its C and C++ counterparts. During the development of the U.S. Step 1 program, there was only one vendor that offered a development suite (compiler, debugger, etc.) for

Ada 95. When problems, or suspected problems, occurred with the development suite, there were no alternative software tools to use. This was especially frustrating when the team was in the process of optimizing the software. There were no other compilers against which the efficiency of compilation could be compared.

***Interfaces are not guaranteed compatible.***

One of the primary advantages of using COTS is the adherence of industry standard interfaces. However, often a large integration effort is required—in some cases, larger than would typically be the case for custom solutions. The development team found that the compatibility of vendors' products depended heavily upon their implementation. For example, a Single Board Computer (SBC) drives the workstation displays with graphics support via a dedicated graphics accelerator. During the selection process, several vendors offered both products, but the team's assessment showed that the best performing SBC and best performing graphics card were offered by different manufacturers. The team did not see this as a problem since their electrical and mechanical interfaces were dictated and followed by industry standards. However, when assembled together in their intended configuration, the combination SBC-graphics card did not work. The compatibility problems were eventually resolved, but it involved intense and diligent efforts over a six-week period by a large team of engineers. Each vendor's implementation of the industry standards was slightly different, causing the unintentional conflicts.

***Consider a COTS subsystem approach.***

Choosing multiple vendors to supply common parts is good for competition. However, any advantage can quickly be erased by increased costs due to additional integration and debugging efforts. By choosing a solution based on a developed and tested product subsystem (for example, the SBC-graphics accelerator pair), the majority of the risk of integrating the subsystem is assumed by the vendor. In choosing this path, the AWACS team would likely save significant schedule and cost.

***Model subsystems to understand systems.***

Early in the design process, decisions must be made regarding the capabilities and performance of various components (i.e., computational efficiency, LAN bandwidth, etc.). Although product decisions are usually made after a system design is complete, our experience shows that a system model in early stages of development would have had a significant benefit in product selection, system architecture, and overall cost and schedule. Unfortunately, with most new COTS products, functional models (including performance information) do not exist. In addition, since the development team does not have insight into the internal architecture of the various COTS components, it is very difficult for them to develop their own models for such components. This modeling is critical to prove out the design. It is worth the additional time required upfront to either work with the vendors to help develop models or their components, or take the additional time required to measure the COTS product's performance in critical areas and characterize it.

***Thorough market research is important.***

Adaptation to the COTS world means market research is required—sometimes extensively. One of the problems with leading-edge COTS products is the lack of good, objective benchmark data. It often is inadequate or does not exist. This makes the selection process much more difficult. It would be wise to develop a set of benchmark programs that exercise all critical attributes of the desired system, and independently run those on products under consideration. Only then can one vendor's product be fairly judged against another. During development of the U.S. Step 1 program, this type of assessment would have likely led to the choice of alternate components.

An example involves the choice of Fibre Channel as the LAN for the U.S. Step 1 network architecture. When the decision was made, it appeared that Fibre Channel was making serious inroads into the LAN market, and would fit the performance requirements dictated by the U.S. Step 1 architecture. Unfortunately, a key requirement—the need to operate in Class 1 Mode (dedicated data transfer path)[3]—was available through only one vendor, which decided to exit the Fibre Channel market in the middle of the U.S. Step 1 development effort. More extensive research, coupled with asking the right questions, may have averted some problems experienced on the AWACS program.

***Choose vendor as carefully as technology.***

All COTS vendors are not alike, and their commitment to the program and their defense contractor partners vary. For example, some vendors excel at providing timely technical support, assist in troubleshooting problems, and provide early insight into new products. Others have a take-it-or-leave-it approach. Based on more than three years of working with various vendors, the AWACS team developed a list of highly desirable qualities for COTS vendors.

1. Choose vendors that have prior military experience. Those with some experience working with defense contractors were more likely to understand the unique customer requirements. They also were more likely to support the program for the long haul, as this is typical of existing defense programs.
2. Choose vendors that want to be on the team. For high volume manufacturers, the AWACS business represents a small portion of their market, so it is often difficult to receive adequate technical support on a timely basis.
3. Choose vendors that are committed to the market. Experience showed that some vendors would quickly change their business strategy regardless of any existing customer commit ments. It is best to try and select a vendor who plans on staying in their current market for the foreseeable future (if they are willing to share their corporate vision with you).

***Partner with COTS vendors, meet regularly.***

A formal face-to-face meeting on a regular basis with the contractors, vendors, and government provides a great opportunity to exchange ideas, concerns, and future plans by the government and vendor, etc. It gives all parties a comfortable sense of their current and future role in the program. It was during such meetings that the Fibre Channel vendor discussed their intent to exit the Fibre Channel switch market. Similar sessions with the other vendors allowed the government, prime contractors, and vendors to express their concerns on every issue ranging from manufacturing capability and schedules to future roadmaps for continued AWACS modernization.

***A careful approach to DMS is required.***

The use of COTS provides many technological advantages, but it also introduces new challenges. One of the most challenging tasks is that of logistical support of the fielded system. For example, numerous versions of COTS products may be identified with the same part number, but it is likely that the exact configuration of parts is not identical. One of the great advantages of using COTS is that you do not necessarily care about what is on the inside, as long as the device performs satisfactorily. Unfortunately, it has been determined from experience that this is not always the case, especially with respect to firmware. Since COTS technology changes at a very rapid pace, a philosophy must be developed that allows an affordable, flexible process with the potential to grow as COTS technology progresses.

***Proactively plan technology insertion.***

Although not implemented, the preliminary strategy planned for the AWACS computer modernization is centered on planned technology insertion. For example, the installation of production kits into operational aircraft was scheduled to take between five and six years. As a result, it is very likely that unless a lifetime buy of production hardware and software was made, different components would be installed within the fleet of 32 aircraft. Since a lifetime buy was not desirable from either a technology or cost-effectiveness point of view, the AWACS team needed to devise a strategy to address this issue. This strategy involved appointing a single manager take charge of tracking COTS, and making recommendations on product choices. This is a multifaceted project involving a number of specialized areas: market research, hardware evaluation, software evaluation, systems integration, etc. The manager would maintain a laboratory and cadre of people dedicated to tracking product changes and improvements, assessing new technologies and market trends. They would evaluate new hardware and software or updated version/releases in the system integration laboratory to determine compatibility, qualification testing, etc. Since COTS design information is generally proprietary, vendors are hesitant to provide detailed design data for performance assessment or debugging. Properly managing this effort requires working closely with manufacturers and vendors, and involves using nondisclosure agreements and other legal and managerial arrangements to ensure the project manager was well informed of any projected changes to the manufacturer's product line. Final approval for introducing new products/technology would involve the use of a configuration control board process specifically designed to address COTS integration.

## Conclusion

The E-3 AWACS System Program Office attempted to incorporate COTS technology into a legacy weapon system. A brief overview of the existing mission computing system and upgrade program were given, along with specific considerations associated with COTS equipment on the E-3. Several challenges arose during development, including:

- Building user support for the COTS approach.
- Establishing baseline architecture with specific COTS components.
- Working with defense contractors transitioning from proprietary practices to the open systems approach.
- Availability of robust software development environments.
- The advantages of using entire COTS subsystems.
- Compatibility problems associated with commercially accepted interfaces.
- The need for modeling COTS behavior.
- Rapidly changing market dynamics.
- Choosing the right vendors.

Although this upgrade program did not lead to production, many valuable lessons were learned.◆

### Acknowledgements

I would like to thank Thad Russell, MEI Technology; Elise Locker, Enterprise Systems Incorporated; and Joseph Bradley, Enterprise Systems Inc. for reviewing the original draft of this paper and providing comments.

## References

1. Orfali, Robert; Harkey, Dan; Edwards, Jeri, *Essential Client/Server Survival Guide*, John Wiley & Sons, New York, 1994.
2. *DSI User's Manual*, Lockheed Martin Federal Systems—Owego, N.Y., 1997.
3. Sohal, Vik and Bunnel, Mitch, A Real OS for Real Time, *Byte*, vol. 21, no. 9, September 1996, pp. 51-52.
4. Milligan, Michael K., *U.S. Mission Computing Modernization Program 1998*.

## Notes

1. POSIX is an industry standard that defines the interface between program code and the operating system. By adhering to the POSIX standard, program code is more portable—thereby avoiding many dependencies on specific operating systems or target hardware.
2. An additional constraint was the long deployment cycle into the field, since only about five aircraft (out of a fleet of 32) could be modified with U.S. Step 1 in any one year.
3. Fibre Channel defines three different classes of service—1, 2, and 3. Class 1 is a circuit switched connection in which an end-to-end data patch must be established before any data transfer can begin. When two devices are using Class 1, that path is dedicated to those devices and is not available otherwise. As a result, access (times) between those devices are guaranteed without any unpredictable interruptions. Class 2 and 3 are also switched services, but without first establishing dedicated paths. In the Class 2 and 3 configurations, data may flow between any two nodes, but the physical path is unknown (and therefore transfer times not predictable).

## About the Author

**Lt. Col. Michael K. J. Milligan** is an assistant professor of electrical engineering at the U.S. Air Force Academy, Col. He previously served as lead engineer and program manager of the AWACS U.S. Step 1 Computer Modernization Program at Hanscom AFB, Mass. He holds a doctorate degree from the University of Texas at Austin, a master of science degree in electrical engineering degree from the University of Massachusetts-Lowell, and a master's degree of business administration from Western New England College. He also has a bachelor of science degree in electrical engineering from Michigan State University. His primary research interests include high-performance computer architecture and real-time systems.

Lt. Col. Michael K.J. Milligan
Department of Electrical Engineering
USAFA/DFEE
2354 Fairchild Hall, Suite 2F6
U.S. Air Force Academy, Colo. 80840
Voice: 719-333-6766 DSN 333-6766
Fax: 719-333-3756, DSN 333-3756
E-mail: michael.milligan@usafa.af.mil

# Creating an Integrated CMM for Systems *and* Software Engineering

By Mike Phillips and Sandy Shrum
*Software Engineering Institute*

*The Office of the Secretary of Defense (OSD) joined with the National Defense Industrial Association (NDIA) in 1997 to sponsor the creation of a Capability Maturity Model®-Integrated (CMMI℠). It would bring together the best features of the Software Capability Maturity Model (SW-CMM) Version 2.0 Draft C, the Systems Engineering Capability Model (SECM) Electronic Industries Association Interim Standard 731 (EIA/IS 731), and the Integrated Product Development Capability Maturity Model (IPD-CMM v0.98). This article describes the CMMI Product Suite that has resulted and what it is designed to provide to the engineering community for enterprise-wide process improvement.*

In 1997 the OSD joined with the NDIA to initiate a project that would integrate process improvement models that would build on the success of the Software Engineering Institute's (SEI's) Software CMM. The Software CMM began as the SEI's answer to a challenge by the Air Force to find a set of key questions about a company's software processes that would guide their selection of the most competent—or mature—software developer. Over several years, that set of questions grew to become the now familiar SW-CMM v1.1.

The systems engineering community had created two models for improvement: the SE-CMM by Enterprise Process Improvement Collaboration (EPIC) and the Systems Engineering Capability Assessment Model (SECAM) by the International Council on Systems Engineering (INCOSE). Those models were later merged successfully to form EIA/IS 731. The two CMMI sponsors agreed that CMM integration would enable the software and systems engineering communities to capitalize on the similarities of their approaches to product engineering process improvement. It also would eliminate some of the differences between the models that had increased the effort—and expense—required to pursue improvement with stovepipe models.

A steering group was formed in 1998. The OSD provided representatives from each of the military services and from the FAA to represent the government. NDIA provided four senior members to represent industry and the SEI provided two managers to round out the team.

Jack Ferguson of the SEI led the Product Development Team (PDT) that was populated with representatives from government, industry, and the SEI. Initially the PDT lacked commercial and international industry participation, but that was remedied later that year. An initial determination to exclude providers of CMM assessments and training because of potential conflicts of interest remained until the draft material was released for public review in August 1999. Since then, participation has been open to these providers.

The initial concept of the CMMI project was that integrating the three source models—SW-CMM v2.0 draft C, EIA/IS-731, and IPD-CMM draft v0.98—would involve little more than combining the practices of the three into a single document. That led to the expectation that the project could be completed in six months. Project members expected that some challenges would result from the differences in scope and life cycle that the models represented and the different approaches to model coverage. However, the team soon found that true integration meant that they had to deal with differences of terminology and model construction as well. EIA/IS-731, for example, contains a large number of practices that define the systems engineering environment for product development. These practices are all considered normative because they must be performed by the engineering organization. The SW-CMM, however, contains a mix of normative and informative elements, including activities and illustrative information (e.g., subpractices).

Compromises were required in order to integrate these two models. The CMMI-SE/SW that was released for public review reflects the efforts to find common ground.

Probably the project team's most controversial decision was to maintain two representations of each model. Early efforts moved toward a hybrid approach similar to the FAA's choice in the iCMM product. However, the systems engineering and software engineering communities, which had lived with two very different architectures for the two models, thought it best to maintain a continuous approach from the EIA/IS 731 heritage and the staged approach familiar to SW-CMM users.

## Challenges Ahead

Releasing the first version of the combined model this summer does not end the development effort. This release was intended to provide an initial operational capability (IOC) for organizations ready to begin the evolution to a more broadly capable model for enterprise process. We will seek feedback from early adopter organizations to gather the lessons learned from using version 1.0 of the CMMI models, as well as the return on investment experienced from institutionalizing the more robust practices that populate the combined model.

Adapting the model to address the particular challenges of working in the integrated product and process development (IPPD) environment has led to a variant (CMMI-SE/SW/IPPD) that better meets the needs of organizations in that domain. Initial work to add the acquisition discipline into the model is under way as well.

While the IOC version of the CMMI models will provide the needed markers for beginning the transition from source models to the CMMI models, we know that the model is still in its youth. Until Lead Assessors have had an opportunity to wrestle with the assessment differences between the source models and the CMMI models; we anticipate differences in interpretation, particularly on the more revised practices. As a result, we advise government organizations seeking to use this latest model for assessing the capabilities of potential suppliers to advance carefully.

Experience also will instruct us as to how to best address the always-contentious area of "tailoring criteria." The current text adds new information to clarify tailoring as an activity at the project level, working with the organizational standard processes. Historically, the challenge has been seen as organizations wrestle with the applicability of various process areas as they seek to demonstrate a maturity level. We expect organizations to give us feedback to get user perspectives on the subject.

We think that the model is in excellent condition for its intended role as a tool to stimulate enterprise-wide process improvement. Nevertheless, we recognize that there remains a need to use such process tools to benchmark organizational capability or maturity. We expect that refinements from actual use will need to be made to the model, just as refinements were made to the Software CMM when it was introduced. Thus, our plans include a v1.1 in about a year that will be designed to capture needed improvements and recognize the need to evaluate as well as assess—to benchmark for source selection discrimination as well as plan the course toward continuous improvement.

Another new challenge not initially anticipated was to accommodate industry's interest in providing guidance through an acceptable standards approach. That thinking was best represented by creating the SECM as an Interim Standard, using the stakeholder balloting process to meet the requirements of the American National Standards Institute. The recommendation to merge the EIA/IS 731 material into CMMI meant that we must consider offering the CMMI model and method as a potential replacement standard.

That possibility drove the need to develop a new appendix that contains required and exacted practices of CMMI models without the explanatory material familiar to users of the SW-CMM. As experience builds with CMMI-SE/SW v1.0, we expect to introduce CMMI materials for consideration first as a national standard and then possibly internationally.

## Our Approach

While we have highlighted some key outcomes in the history, in this section we would like to give you a feel for the processes—they were not always as "high maturity" as we might have liked. Creating the Product Development Team was a specific attempt to include diverse representatives from government and industry with the SEI. Organizations were asked to provide team members rich in experience with the source models and their training and appraisal methods. In many cases, the members had authored portions of the source models, and were seldom shrinking violets about the strengths and weaknesses of the sources and related reference models and directives. One member noted that in a given meeting, one could estimate the number of opinions by counting the number of members in the room and adding at least one. Decisions that favored one model's features over another always had to be sought by building team consensus. This led to creating the first version, v0.1, which was released to a stakeholder/reviewer group in late 1998. Difficulties were pointed out in the size and complexity of the model, and the team sharpened its pencils to better integrate the material. It was a major step when the team, which had focused on the CMM's engineering portion, determined that the expanded practice areas inherited from the systems engineering discipline could represent a more complete evolution of the significant portion of product development represented by a single Key Process Area in the SW-CMM, Software Product Engineering.

The release of v0.2 for public review at the end of August 1999 started another round of improvement. By the close of the review period in November, there were about 2,500 change requests. In addition, offerings of the draft Introduction to the CMMI course training were coupled with focus group sessions the following day. Workshops in technology change management and high maturity practices provided similar change recommendations for consideration. Each of the pilots designed to exercise the model and the appraisal method also provided opportunities for feedback and improvement. The Editor Team, led by Drs. Dennis Ahern and Mike Konrad, determined which changes could be accommodated in time to maintain a summer delivery of the product suite, and those that would not be pursued or would be deferred for consideration.

The quality of thoughtful comments made the choices difficult. In some cases the need to maintain consistency between continuous and staged representations caused some constructions that appeared confusing to the readers. We sought to clean up these for v1.0, but we know there will be room for improvement.

As v0.2 was beginning the review process, the Undersecretary of Defense (Acquisition, Technology, and Logistics) signed a policy recognizing the value of process maturity for the Department of Defense's developers of software intensive systems. This led to the need to clarify that the CMMI Product Suite was not developed for source selection. We thought that even though the model represented a broader look at engineering development than with any of the earlier source models, the CMMI Product Suite would need maturation to assure comparability of appraisals before it might reasonably be used by the government for evaluative comparison.◆

## About the Authors

**Mike Phillips** is the SEI's Director of Special Projects, a position created to lead the CMMI project for the SEI and the Steering Group. He was previously responsible for Transition Enabling activities at the SEI. Prior to his retirement as a Colonel from the Air Force, he managed the $36 billion development program for the B-2 in the B-2 SPO and commanded the 4950th Test Wing at Wright-Patterson AFB, Ohio. In addition to his bachelor's degree in astronautical engineering from the Air Force Academy, Phillips has master's degrees in nuclear engineering from Georgia Tech, in systems management from the University of Southern California, and in international affairs from Salve Regina College and the Naval War College.

**Sandy Shrum** is a member of the CMMI product-development team and has been a senior writer/editor at the SEI since 1995. Before joining the SEI, she spent eight years with Legent Corp., where she was a senior information developer, a member of a software-development team, and a member of Legent's Information Technology organization. She has a master's degree in professional writing from Carnegie Mellon University and a bachelor's degree in business administration and marketing from Gannon University.

Software Engineering Institute
Pittsburgh, Pa. 15213-3890
Voice: 412-268-5800
E-mail: customer-relations@sei.cmu.edu
Internet: www.sei.cmu.edu/cmm/cmms/cmms.integration.html

# The Demarcation Zone: Surviving a CMM Assessment

By Deb Jacobs
*Priority Technologies Inc.*

*A much ignored subject is the preparation for and conduct of CMM®-Based Appraisal for Internal Process Improvement (CBA-IPI) for an organization, referred to in this article as a CMM assessment. That awareness was emphasized while conducting a mini-assessment. There were many weaknesses identified for the organization, but the underlying reason for these weaknesses and inaction to correct them rested directly on the Site Coordinator and the team preparing for the assessment. This team is critical to the success of an assessment. These are the people that are at the line of demarcation or on the front lines fighting to make the organization the best it can possibly be.*

When an organization prepares for and conducts a SW-CMM CBA IPI-type of assessment, there must be someone in the driver's seat. Each organization has definitions and roles defined for its process improvement efforts, including a Site Coordinator or lead process engineer. This is the person typically behind the steering wheel.

This role will be referred to as the Site Coordinator throughout this article. Each organization's terminology may differ but the intent is the same. The Site Coordinator is responsible for leading a team of organizational staff members in preparing for an upcoming CMM assessment.

## External vs. Internal Assessments

The rules under which a CMM assessment is conducted are open to tailoring by each organization. Some choose to conduct their assessment using a team composed of mostly internal members of the organization, others prefer using members who are outside of the organization. The Software Engineering Institute's (SEI's) Technical Report concerning CBA-IPI [1] states several alternatives to building an assessment team, "as one team member is from the organization being assessed."

There are advantages and disadvantages to using a method where most of the team is external or most of the team is internal. The following figure lists some of the advantages achieved from both methods. An organization must decide which of these are important to its organizational goals.

It is critical that an organization weighs the pros and cons of each approach. They will vary with each organization. The key is to recognize the organization's true culture.

Some argue that a strong assessment lead, Process Asset Library, and organized evidence with a well-versed staff will alleviate many of these problems. That is true to a degree, Even when following a model such as the SW-CMM an organization can implement the various elements in a multitude of different ways. There is only so much time to conduct an organization's assessment. There is not enough time for one or two people to fully understand the organization in order to adequately advise as to how to improve that organization's processes. A good process mentor who works with the organization closely building up to the assessment would help, but he or she must be careful not to become so close to the organization that the mentor overlooks something.

There is a potential for a software capability evaluation (SCE) for organizations working government contracts. In that case, an organization would rely on an external team of assessors to ensure preparedness for a potential SCE.

Most of this author's experience is based on using an external team; this article is written with that premise in mind.

## What a Site Coordinator Does

The Site Coordinator is responsible for leading a team of organizational staff members in preparing for an upcoming CMM assessment. This team is typically called the SEPG. For one organization, it was called the Process Group since responsibilities included more than software and engineering. It included all of the processes by which the organization operated or at least a tailored version of the overall organization's processes. The Site Coordinator, along with one or more other team members, normally participates as a member of the assessment team.

**Advantages of INTERNAL Team**

- Lessons learned from the experience are invaluable
- Able to fully understand what needs to be done to correct problems encountered
- Less costly than bringing in external members
- Less coordination required to bring the team together
- Decreased preparation time for assessment since members familiar with the organization and the process
- More trained staff members in assessing the organization

**Advantages of EXTERNAL Team**

- No preconceived notions concerning the organization and normally, few or no preconceived notions concerning implementation of the model
- Complete independence from organization - no vested interest or threat of negative consequences based on results of the assessment
- Ability to better prepare the organization for outside assessment by the customer (e.g. SCEs)
- For a service organization, provides more credibility to findings in many customer's view
- Can provide the organization with several best practice perspectives based on their various areas of their expertise
- Many times internal members are too close to the problems to see them where an external team would much more likely be able to readily spot problems

The Process Group is the team responsible for all aspects of the assessment, from helping the organization prepare for the assessment to working with the assessment team to ensure that it understands how the organization operates based on the SW-

CMM. The Site Coordinator and designated team members prepare the engineers and managers for interviews, as well as the reams of evidence to verify an organization's CMM-related capabilities in addition to other logistical tasks needed for conducting the assessment.

The Site Coordinator and designated team members will explain, define, and at times defend the organization to the assessment team. This is especially true when evidence or processes are not apparent or well understood. In an ideal world, this would not be necessary but experience tells us that the world is not ideal.

The SEI for CMM assessments specifically defines a Site Coordinator's role. The SEI's Technical Report concerning CBA-IPI describes it as:

*"The individual responsible for handling the logistics of the assessment. The site coordinator is responsible for developing the schedule, notifying the assessment participants of the schedule, making sure that adequate rooms have been reserved for both the pre-onsite and onsite periods, making and distributing copies of the schedules, making sure that all necessary supplies and equipment are available when needed, scheduling contingency interviews, requesting additional documentation, and ensuring that meals are taken care of. The site coordinator needs to be a member of the assessment team [1]."*

This description is accurate for most organizations, but many times the role entails much more that this addresses. According to Ken Dymond in *The Assessment Coordinator's Handbook*, "The Site Coordinator is the person whose efforts are invisible if the assessment goes well and the person blamed if even the smallest planning item, in all the closely scheduled moments of the on-site period, is wrong. Success is transparent, but failure stands out [2]." This is so true. There is so much information to prepare but even the most minute of details can get you in trouble.

Collecting artifacts, also known as evidence, is a daunting task. Many times an organization takes it too lightly. An organization may be fully prepared and in line with all aspects of the SW-CMM Level it is targeting but it must be able to demonstrate that to others—an assessment team. Demonstrating it to others is the tricky part.

## Importance of the Site Coordinator's Role

There are numerous key roles in attaining process maturity. These include, but are not limited to senior management, sponsors, middle management, assessment team lead, process improvement mentors, and the Site Coordinator.

In preparing for and during the assessment, especially when most of the team members are external, the Site Coordinator and designated process group team members can make or break an assessment. It is extremely difficult for four to 10 people unfamiliar with an organization to come in and assess where they stand. The Site Coordinator is key to helping them understand how an organization fits into the SW-CMM framework. They must be able to explain to an assessment team how an organization is compliant with a method that is open to many different interpretations due to its very nature.

The SW-CMM was written to provide a guide for an organization, not to tell them how to do it. It describes the characteristics of successful software processes; hence, the varied ways of implementing the SW-CMM are infinite. Guiding the assessment team in understanding the organization's way of doing business is critical. Some assumptions the team makes without that guidance may not be accurate, due to a lack of understanding the organization's culture.

## Makeup of a Successful Site Coordinator

Mark Paulk, of the SEI, listed several proverbs in his article *Software Process Proverbs,* including "Competence: The competence of the people who do the work is crucial to project performance and organizational success [3]." We have all heard the saying "one bad apple spoils the batch." That works the other way as well. One good apple can bring more to an organization than a dozen without the proper attributes and skills.

A critical task an organization will face when attempting process improvement is selecting the right person as the Site Coordinator. It is not easy to find people with the attributes necessary to be a successful Site Coordinator. This is not intended to discredit the tried and tested CBA IPI assessment approach, which is valuable. However, this author has seen how an unprepared, unorganized organization can fail with even the most experienced assessment team trying to help the organization improve.

It takes a special attitude to be a successful Site Coordinator, so organizations must be cognizant of whom they select. Conversely, the person must be sure that he or she is up to the challenge, since it will mean many long, sleepless nights and weekends. The frustration level will be so great at times that they will want to throw their hands in the air and leave. All of that frustration is worth it, but the Site Coordinator must understand what he or she is "volunteering" to do.

There are many attributes that make for a good Site Coordinator, but what is good for one organization and assessment team may not be for another. The following table lists the most critical attributes necessary for a successful Site Coordinator, but there may be many more depending upon an organization's unique circumstances. These attributes also can encompass an entire team rather than one person.

**Makeup of a Successful Site Coordinator**

- Confident without huge ego.
- Proactive and willing to take reasonable chances when necessary.
- Organized/ability to organize without over-organizing (avoid bureaucracy).
- Visible, respected member of the organization.
- Easygoing but not so much that things do not get done.
- Ability to communicate at all levels, from managers to practitioners.
- Ability to interpret explanations made by both management and practitioners (assessment team to interviewee(s) and interviewee(s) to assessment team).
- Nonargumentative.
- Detail oriented.
- Good listener, open to others' opinions.
- Strong without being overbearing.
- Willing to bend when necessary, knowing when it is necessary.

- Totally committed to process improvement, the organization.
- Immense drive/motivation and ability to drive others.
- Ability to work in stressful situations.
- Able to withstand criticism from levels internal and external to the organization.
- Ability to keep a proper perspective (does not get angry).
- Focused Adaptable—be able to roll with the punches.
- Never lose sight of goals.
- Creditability with senior management [1] and, even more importantly, with practitioners.
- Ability to maintain confidentiality.
- Ability to lead and follow, depending on circumstances (must recognize when each is appropriate).
- Ability to become an expert in many different areas not their normal field of expertise—may have to be self-taught.

If an organization finds a person with many of these attributes, the rest will be learned along the way if an organization is to be successful. This person must be committed, proactive, open minded, and hardworking with the appropriate authority to make things happen. It will be a wonderful, irreplaceable learning experience.

## Sharing Site Coordinator Responsibilities

It may be necessary to have more than one Site Coordinator to share the responsibilities, depending on the organization's size and the assessment's scope. It is absolutely necessary that these individuals are compatible. They must work cohesively. This was a glaring error this author discovered while performing a mini-assessment of another organization. There was so much dissention between the co-Site Coordinators as well as the entire team preparing for the assessment that it was amazing that they proceeded with the formal assessment.

Not surprising, the result was SW-CMM Level 1. Much of this was due to the team preparing for the assessment, especially the Site Coordinators. Even though they shared a common goal, their methods of achieving it were diametrically opposed. It is critical that the process engineering team is synergistic. It does not matter who is right or wrong, only that the team communicates openly and cordially with a common vision.

## What a Site Coordinator Must Know

The Site Coordinator's knowledge base grows increasingly large as he or she prepares for an assessment. Since that person will be in the line of fire, he or she must become expert or close to expert, with as many elements as possible within the organization. There will normally be others to search out for specific answers but the Site Coordinator must know whom else to seek out when necessary.

The site coordinator must: know every piece of paper in every file, know where the bodies are buried, know an organization inside and out, know and fully understand SW-CMM and as many interpretations of it as possible, be able to explain, and, when necessary defend, evidence without being argumentative, be able to explain and, when necessary, defend the organization's programs methods without being argumentative. Normally, the Site Coordinator—along with the process team—is the one who collects or leads the collection of evidence. This is not always the case, so he must know all of the evidence and how it applies to the SW-CMM. This will enable the Site Coordinator to understand the organization's method of meeting each practice for the SW-CMM.

It is equally important to understand the SW-CMM and its many interpretations, since by its very nature it is open for interpretation. Many times an organization will have alternate practices that meet the intent of a SW-CMM practice. The Site Coordinator must understand these and how they fit. There will be wording and terminology differences between SW-CMM, the assessment team members, and the organization. Terms must be clearly explained to the assessment team, including how they relate to the SW-CMM—the Site Coordinator sometimes needs to tie it together for the team.

## The Assessment or the Organization: What is Your Real Role?

As an assessment team member, the Site Coordinator and any other internally designated assessment team members must be able to objectively judge the organization as an outsider. This can be difficult since there may be so much at stake for those involved. With government contracts, a future or imminent project may depend upon an assessment's outcome.

Even when not the case, an organization invests a great deal of money in process improvement and fully expect to see successful results. Many times reputations and future opportunities are based upon the outcome of a CMM Assessment. One Site Coordinator said that his next promotion depended upon the results of the CMM Assessment.

The Site Coordinator has an obligation to the organization. It is essential to strike a balance between these two goals. The Site Coordinator must be able to act objectively as an assessment team member and as part of the organization at the same time. What a challenge! There are many human aspects that you must work out. It can be challenging, exhilarating, and painful at the same time.

## Bottom Line

It is easy to underestimate how difficult it is to attain a specific process maturity level, regardless of the methodology selected. It is a daunting undertaking. As one who has been at the demarcation zone, it may be the most challenging thing undertaken in a person's entire career.

If asked to be a Site Coordinator, ask yourself what it takes to push your comfort level. Remember the satisfaction and rewards gained are well worth any pain endured.◆

## References

1. Dunaway, Donna K., and Masters, Steve. Technical Report CMU/SEI-96-TR-007 ESC-TR-96-007, CMM[SM]-Based Appraisal for Internal Process Improvement (CBA-IPI): Method Description, pp. 7-8, 14, 38.
2. Dymond, Kenneth M., *Assessment Coordinator's Handbook: Planning for a Well-Orchestrated Software Appraisal*, Process Transition International Inc., 1997.
3. Paulk, Mark C., Software Process Proverbs, **CROSSTALK**, January 1997.

## About the Author

**Deb Jacobs** leads the Project Engineering Management Group at Priority Technologies. She has 26 years of experience working in the Information Technology industry. She began her career in the Air Force as a technician working with computers on B-52's and KC-135s. After completing her bachelor's degree in computer science she continued working with the information technology industry in a broad range of areas, including software engineering, process engineering, and project management. One of her most notable successes was leading the team responsible for achievement a rare CMM Level 3 rating in record time. Jacobs is chairwoman for the CERT Conference Committee; is newsletter editor/originator of the Omaha SPIN Newsletter, SPINOUT; and works with the SEI on the integrated Capability Maturity Model.

Priority Technologies Inc.
1508 JF Kennedy Drive, Suite 101
Bellevue, Neb. 68005
Voice: 292-1212
Fax: 292-1215
E-mail: djacobs@prioritytech.com



# Incredible Suckers

After wrestling for supreme control of the remote the other night, I sat down for some nightly entertainment with my son Matthew. Jumping from channel to channel we searched for a program we both liked. He tossed out "Behind the Music" and I eliminated "The Wild Thornberrys." He scoffed at "SportsCenter" and I vetoed "Doug." We both gagged on "Friends." Finally we hit PBS as the host of "Nature" introduced the program:

*"A decade of discoveries has revealed the extraordinary possibility that the prime intelligences in the ocean may not be the swimming mammals (whales and dolphins), but instead a race of "incredible suckers, the cephalopods."*

A bag of pretzels, a drink, and suckers . . . count me in. Venturing into the world of chambered nautilus, cuttlefish, octopus, and squid we had found a common interest—gooey squishy things that think.

Halfway through the program and the bag of pretzels, a marine biologist introduced the blue ring octopus, one of the smallest but most deadly of the cephalopods. There was a ring of familiarity as he described the effects of the blue ring's bite.

"*The bite of the blue ring is not much of a bite at all. In most cases you don't even know you have been bitten. It is rare to find a puncture wound or the site of the wound. It is almost like they force or inject toxin through the skin. However it administers the toxin, it is effective.*

*The symptoms of the bite of the blue ring octopus are incredibly consistent. First you feel a kind of numbness around your mouth and lips. Then it becomes very difficult to breathe, followed by a general paralysis of your body. Your knees wobble, you collapse, and you lay on the ground with your eyes fixed and dilated, totally unresponsive to everything around you. You cannot move a muscle. The weird thing, though, is that your mind remains relatively clear. You can hear, understand, and remember what everyone around you is saying. It probably does not help to hear things like, 'that chap's had it.'"*

Having never encountered a blue ring octopus, I was curious as to why the symptoms of the blue ring bite were so familiar. Unsuspecting bite, numbness around the mouth, difficult to breathe, paralysis, eyes fixed and dilated, yet your mind remains clear? Staff meeting!

That's it, he's describing the symptoms of a staff meeting—not just any staff meeting but the dreaded Blue Ring Meeting (BRM). The BRM is a gathering with no purpose, leadership, control nor participation. A rendezvous in which otiose information, that could be sent in a quick e-mail message, is stretched over hours of excruciating tedium. A parley for colleagues to out-feign each other's interest in such tedium. It is an event in which you commonly sight lockjaw, doodling, drooling, and the amazing vertical snooze.

They appear to be like any other meeting. You seldom feel the sting, but once bitten it is over. Your mind is clear but you can not move a muscle. Thoughts run rampant. You devise little games to maintain a sense of coherent understanding but it is a losing battle. Try to take some action or cut the meeting short and the BRM embraces you with its tentacles and sucks the life out of you.

If you survive the dreaded BRM, which most of us do, there is one terrible side effect. You start holding your own Blue Ring Meetings. Even though you despised the experience, the helplessness, and the pain you pass the experience of the BRM on. It is a vicious cycle. It is a horror of all horrors. BRMs propagate themselves.

How can we stop this malignant tryst? Many have tried, and there is a plethora of books, videos, and seminars on the subject. Maybe we should organize. Join SWEABRM—Software Engineers Against Blue Ring Meetings. We could organize a Million Engineer March. My advice, when your boss asks you how the meeting went, answer: "Incredible Sucker."

*—Gary Petersen, Shim Enterprise Inc.*

***CrossTalk / TISE***

5851 F Ave.

Bldg. 849, Rm B04

Hill AFB, UT 84056-5713

**PRSRT STD**

**U.S. POSTAGE PAID**

**Kansas City, MO**

**Permit 34**