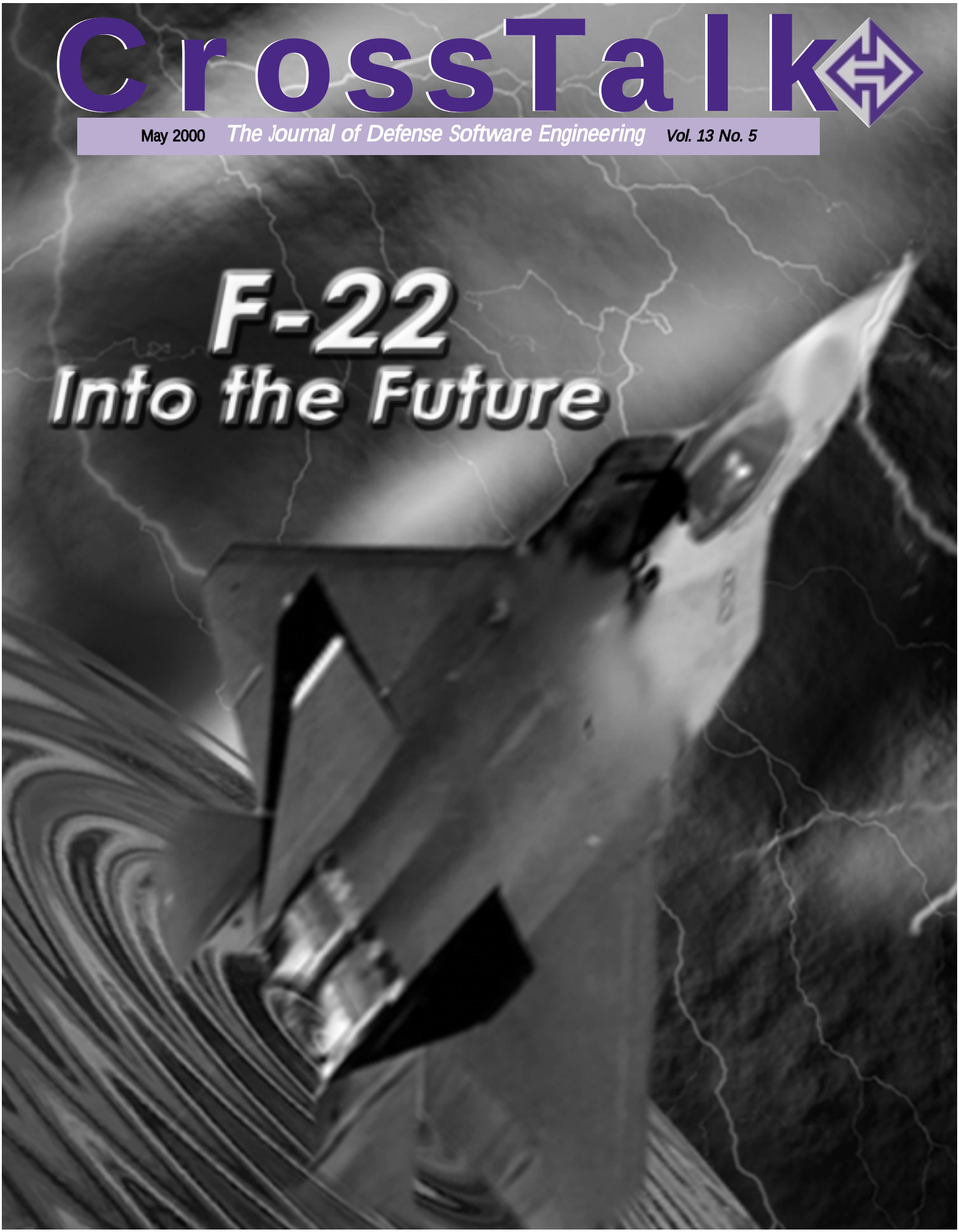
CrossTalk

May 2000 The Journal of Defense Software Engineering Vol. 13 No. 5



The F-22

About the Artist

Brandon Scott is a graduate of Utah Career College where he studied the latest multimedia techniques and software programs. He enjoys everything from abstract design to web design.

Software Engineering Technology

Open Forum

Departments

CrossTalk

Sponsor	H. Bruce Allgood
Publisher	Reuel S. Alder
Associate Publisher	Lynn Silver
Managing Editor	Kathy Gurchiek
Associate Editor/Layout	Matthew Welker
Associate Editor/Features	Heather Winward
Voice	801-775-5555
Fax	801-777-8069
E-mail	crosstalk.staff@hill.af.mil
STSC Online	http://www.stsc.hill.af.mil
CrossTalk Online	http://www.stsc.hill.af.mil/Crosstalk/crostalk.html
CRSIP Online	http://www.crsip.hill.af.mil

Subscriptions : Send correspondence concerning subscriptions and changes of address to the following address:
Ogden ALC/TISE
7278 Fourth Street
Hill AFB, Utah 84056-5205

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 http://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.

Study Success, Learn from Failure

We are always looking to improve the quality of information in **CROSSTALK**. We rely upon articles that we run through a technical and editorial review process. This has served us well, but is no substitute for a view from the senior leadership upon whom we all depend for vision and direction. Reader surveys of the past have shown us that the vision and direction of senior leaders is highly desired. In fact, over the last year one of the top 10 articles as measured by our Web hits was from Dr. Patricia Sanders, the director of test, systems engineering, and evaluation for the Department of Defense.

With that vision in mind, I approached Brig. Gen. Michael Mushala. His gracious response resulted in interviews with Maj. Gen. Claude M. Bolton Jr., Office of the Assistant Secretary of the Air Force for Acquisition, Washington, D.C.; Thomas C. Brandt, Associate Director at the Software Engineering Institute Carnegie Mellon University; and Ronald D. Dubbs, F-22 Weapon Systems Chief Engineer, Wright-Patterson Air Force Base, Ohio. The result of these interviews begins on page 4.

What stood out at each of these interviews was the outstanding job that the F-22 program has done with its software. The F-22's productivity numbers were impressive. Those numbers impressed Dr. Dave Cook of Draper Laboratories, who serves on the STSC consulting staff and acted as **CROSSTALK**'s technical advisor on these interviews. The work on the F-22 was done with a geographically and organizationally disperse community of software developers. In addition, F-22 technical and managerial prowess is articulated in detail by a recently completed SEI risk mitigation intervention that Maj. Gen. (Ret.) Brandt shared with **CROSSTALK**. Brandt was lavish in his praise of the F-22 leadership.

Can everyone's software be as successful as an F-22 program? The laws of statistics may challenge us on this, but I believe we need to study success and learn from failure. I also had my eyes opened to the wealth of information that the F-22 has to offer. Clearly the F-22 did a lot of things right. A case study of the process, methods, tools, and environment would yield significant material for study and a broader range of future application. However, the people doing the work often do not have time to report on their work and frequently are too close to their work to understand the significance of what they have accomplished. The **CROSSTALK** staff looks forward to doing more of this kind of information gathering and to reach deeper into the valuable lessons learned.

CROSSTALK brings several special elements to readers in this issue, in addition to the aforementioned interviews: an article on page 12 by Beverly Moody, the Avionics Software Block Lead for the F-22; and a special pullout—an F-22 Raptor poster—created by graphic artist Mark Driscoll of Salt Lake City, Utah.

Reuel Alder, Publisher
Software Technology Support Center

Up Close with Maj. Gen. Claude Bolton Jr.

Maj. Gen. Claude M. Bolton Jr. is the program executive officer for fighter and bomber programs for the Office of the Assistant Secretary of the Air Force for Acquisition, Washington, D.C. His responsibilities include acquisition activities on the F-22, F-15, F-16, F-117, B-1, and B-2 programs and the Common Missile Warning System and Joint Helmet Mounted Cueing System programs. He was commissioned through the University of Nebraska's Air Force Reserve Officer Training Corps in 1969. He is a command pilot with more than 2,700 flying hours in more than 30 different aircraft, and flew combat missions in Vietnam.

CROSSTALK: Having seen the F-22 program from its beginning, what actions in the software area could have been taken to improve the current product or prevent problems you have encountered?

Gen. Bolton: We could probably use another metric or two. We had some that we got rid of and we could use a few more of them. If we had to do it over again, we would look at all of our metrics and leave them in there a bit longer until we understood what they were telling us.

CROSSTALK: For F-22-integrated avionics and software, how are you doing and how do you know?

Gen. Bolton: Really well. We have a Monthly Executive Review where we invite the key players from the contractor side and the government side. The purpose is to look at the various issues; one is how we are doing on software. We have a schedule, a block approach that we are taking.

How are we doing on the burn down? What are we accomplishing in the ground-based laboratories? How are we doing in the blocks, coding, verifying, and checking and putting that through the laboratories and so forth? That is how we track it at this level.

At the [Air Force] System Program Office, contractor and the Integrated Device Technology [level], it is a day-to-day thing; they are really into the details. We do not get into that. We do not have the time and we would just muck it up. Right now the biggest thing is the schedule. With everything we have delivered—whether to the Avionics Integration Lab [in Seattle], the flying test bed, to the test birds—over the last nine, 12 months it has been on or ahead of time.

CROSSTALK: You attribute that to these monthly and daily meetings?

Gen. Bolton: That is part of it. Most of it is because of the structure of the avionics program, the block approach[1] that we take, and that we have ground-based labs. We have one in Fort Worth; the big integration unit is in Seattle where all the pieces come together. Once it is all proven there, we put it onto the flying test bed, which is a converted 757.[2] It is all there; we duplicated the cockpit for the F-22. If you look at the flying test bed, you will find the cunard [a sensor part] up front. We have upwards of 30 engineers who look at the data streams and say 'that is what we expected, this is what we are getting, how does that compare to what we have on the ground?'

Once it is run there, we put it into the actual airplane. That process is really what is causing this success. When we look at the schedule, it allows us to ask 'how is it going, what do you need from us?'

CROSSTALK: It sounds as if you had to do it all over again, the block approach and the intensive meetings are both things you would keep. Is there anything that you would do differently?

Gen. Bolton: We put together a team [around December 1998] that emphasized the integration of the team approach. We were looking at R-19. I said, does that mean we have 19 [schedule revisions] since we started research and development? Well, yes . . . every six months we look at it and put another revision on [the schedule].

> "We made the 'i' a capital 'I' in the Integrated Product Team . . . That is paying big dividends."

We needed to be more proactive, rather than reactive and changing the schedule. Let's figure out where we ought to be. Let's put a team together across the enterprise and put the best and the brightest folks on it. We made the 'i' a capital 'I' in the Integrated Product Team (IPT). It brought more folks together. That is paying big dividends.[3] We went to R-20 and held schedule longer than we ever have [April 2, 1999 to present].

We initially were going to fly the Block 3 software in April 2001. The team looked at that. Because of all the things we were doing, because of the confidence we have in the processes we have, because the successes we have had over the last year or so, we believe we can move it from April 17, 2001 to [putting software in the aircraft] around Oct. 30, 2000. It goes back to the team understanding what is going on, and everybody working on the problem—the best way to get this sent to the aircraft and do it safely, professionally, and correctly from an engineering standpoint.

CROSSTALK: What do you believe are the key metrics for F-22 software development?

Gen. Bolton: The schedule and the cost. If it were not for the cost, this would be easy. Inside the [Washington] Beltway, for this program cost is the primary factor. Everyone is concerned about that. The contributors to cost in this program are avionics and airframe. I am very sensitive to how much it is costing, and that sensitivity is shared by the team. From a software standpoint that may not be the answer a lot of folks want, but unfortunately we cannot do the things that need to be done if we do not have the money to do them.

CROSSTALK: Did your system have any unforeseen setbacks?

Gen. Bolton: Early on in Block Ø we had some surprises. That is because we were going from a first-ever design, even though the design tools and practices environment were the best I have ever seen. You put it into some type of ground-based laboratory,

then into an integration laboratory. You say, 'let's go fly this,' and find a flying test bed.

That showed us early on that we had some problems. I would not call them major. It is a whole lot better to find it there, in the ground integration laboratory or the flying test bed. We have found most of our errors on the ground in the integration test laboratory. To give you an example: We put the radar—which is electronic, very complicated, software-intensive—into the flying test bed, turned it on, and it received targets the first go, [which is] unprecedented for radar.

"Our first best practice is the IPT. The next is the use of ground-based laboratories with the flying test bed. I am not sure how we can do this program without the flying test bed."

CROSSTALK: Did your team develop or collect best practices?

Gen. Bolton: Our first best practice is the IPT. The next is the use of ground-based laboratories with the flying test bed. I am not sure how we can do this program without the flying test bed. You have the opportunity to take from the ground and put into a flying environment all the boxes—you have the cockpit, all your engineers there, and folks looking at the data stream. That is beautiful. You can see it all right there. On a flight that can last hours you can get a hell of a lot done, far more than you can on an actual fighter and at a far reduced cost. Those are really key: the IPT, the ground-based laboratory, and the flying test bed.

CROSSTALK: What is your vision for process improvement involving the F-22?

Gen. Bolton: We go back to the team. We are always hammering the cost for this. The team was already ahead of us. There is continuous process improvement.

This process turns out not to be dependent solely on personalities. We have had Dr. Hans Mark [Director of Defense Research and Engineering] take a look at us several times because of his vast experience on other systems, particularly flight controls, avionics, software, and structures. The last time (November 1999) we spent a couple of hours with him and went through excruciating detail.

His bottom line was, 'You are doing the right things.'

Notes

1. See page 10 for comments by Ron Dubbs, F-22 Weapon System Chief Engineer at Wright-Patterson AFB on the usefulness of the block approach to the F-22 program.
2. See page 14 for more on using the flying test bed in the F-22 program.
3. See page 8 for comments by Maj. Gen. (Ret.) Thomas Brandt on the value of the IPTs. See page 9 for Dubbs' comments on the effectiveness of IPTs.

Maj. Gen. Claude Bolton Jr.
AFPEO/FB, 1060 Air Force Pentagon
Washington, DC 20330-1060
Voice: 703-588-7300

Asking the Right Questions

The Air Force had a program, now defunct, called Bold Stroke that was a two-day course for colonels and flight officers.

"The bottom line was to teach them how to ask the right questions," said Maj. Gen. Claude Bolton Jr.

Asking the right questions goes all the way to the top. Bolton cited the example of some software trouble reports he had studied that showed that a particular program was 50 percent complete. Several weeks later, the report showed the same thing: 50 percent complete.

"Wasn't that what it was two weeks ago?" he recalled asking a staff member. 'Yes sir,' the staff member replied with a big smile.

Something was wrong, the general concluded. He told the staffer he wanted the following points addressed regarding the report:

- What does the software trouble report tell me?
- How do I know?
- Show me.

"OK, you have a week. See you," the general told the staff member.

Monday morning came.

The new report showed that the program was 52 percent complete. Now they were getting somewhere, the general remembered thinking. Confident that the staffer knew the answer, the general asked what it meant for that program to be 52 percent complete.

"There was no answer," Bolton recalled. "There were a lot of words that came out, but there was no answer. It took us a month to understand what that meant. And trying to answer that question, we looked at our process in that program and with the contractor and found out that our quality control folks were not in the process. The library where we were supposed to be booking this was out of sequence. And this had been going on long before I got there. We had to get the right expertise in there. Just by going through those questions we were able to turn that program around."

Dwindling software expertise in the military is another factor in managing a software-intensive program.

"When I first got into this part of the business and started what is now the F-22, we knew we were going to have the heart and soul of the avionics," Bolton said.

"The pilot would be a manager of the weapon system. We were going to have a lot of lines of code. An expert tells me he can give me 30 lines of code and cannot test it exhaustively because it takes him a lifetime to do it.

"To that end I do not think we have come very far in the last 17, 18 years. We have put more manpower on writing the lines of code. We have somewhat better tools. But if instead of 1.9 million lines of code, I want a billion lines of code, how do we do that? Can't. And if you say that, I guarantee you that you will never have a Star Trek. You will never have the Enterprise." Bolton said.

"I am of the notion that there is a way of doing that. Software has got to get easier in terms of our ability to create it, test it, and use it.

"Military and industrial software expertise is dwindling on both sides. Our expertise has gone by the wayside and it is not going to improve. It is going to the commercial sector.

"Let us go back and look at what is causing concerns as we develop software. Is it the writing of code? Is it the testing of the code? Is it the integration? Is there a way we can engineer or manage our way out of this so we do not have to have all that expertise?

"We have not stepped up to that."

The next step in software management, Bolton concluded, is answering the question of how to continue putting out a quality product, but with far fewer resources.

Up Close with Maj. Gen. (Ret.) Thomas C. Brandt

Thomas C. Brandt is Associate Director at the Software Engineering Institute (SEI) at Carnegie Mellon University. Appointed to that position in January 1995, he is responsible for SEI activities relating to strategic development initiatives for rapid transition of software technology. Brandt has more than 39 years experience as an engineer and senior executive specializing in the development and operation of high technology space, missile, and command and control systems. He was a test director for both the Atlas and Minuteman programs and participated in more than 40 ballistic missile launches. As an associate professor in the department of astronautics and computer science at the United States Air Force Academy, he helped develop digital-based open form solutions to complex problems in celestial mechanics.He has been associated with the military space program for more than 25 years and was responsible to the Joint Chiefs of Staff as the planner for the United States Space Command. As vice commander of the Electronic Systems Division he had a broad range of responsibilities for the research, development, and acquisition of strategic systems. He is a graduate of the United States Naval Academy at Annapolis, Md., and holds a master's degree in electrical engineering from the Air Force Institute of Technology. He is also a distinguished graduate of the Industrial College of the Armed Forces and the Executive Program of the School of Business at the University of Virginia.

Brandt: The F-22 is, at this stage, a very happy story. That is good to report because the stories that I have recently encountered generally are not in that category. Quite often when people find their program troubled they will ask for someone to come in—like an independent review group, a Tiger Team. In this circumstance you are cast in the role of the pathologist, or the coroner, or even Jack Kevorkian: 'I am here to help you, breathe deeply two times.'

That was not the case for the F-22 review late last June. It was really a risk mitigation intervention, to look at the state of the development [of the F-22 avionics]. This was late in the research and development phase to determine:

- The risks before and during the terminal phase of the Engineering and Manufacturing Development.
- The segue into low-rate initial production.
- The tradeoff between live fly testing, which tends to be very expensive, and modeling and simulation testing as an alternative.

There are two advantages to modeling and simulation, which in this program turn out to be a best practice. Modeling and simulation, with the fidelity we now have in computation, allows testing of the dynamic envelope of a product as it performs. We were focused on the avionics suite that has some unprecedented application of emerging technologies. An example is sensor data fusion—not an easy technical job—component in the avionics tend to be software-intensive, whether it is the inertial reference system or the so-called Communications Navigation and Intelligence suite. The avionics is a federated system of components, all of which must interoperate in highly complex, sophisticated ways. From the software perspective, we have made progress over the last decades. However, the continued evolution of electronics is accompanied by demands for increasing complexity and has masked progress. As our methodologies to solve complex problems have evolved, the problems tend to expand, driven by even greater complexity.

CROSSTALK: What was it about modeling and simulation that made it a best practice? Did you test in real environments?

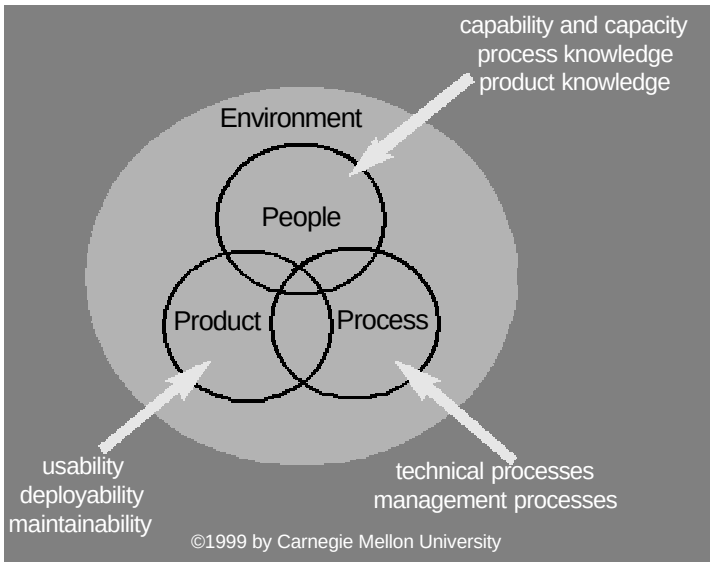
Brandt: There are two aspects that have made it so good—the maturity of the technology now, and the fact that it was employed as a strategy in the program early on and is pervasive. It is not just one model and simulation; it tends to go down and across the tiers. The F-22 is complex and employs a workforce that is creating the product. As you look at these various tiers, there are about 4,500 contractors, so the management challenge is large. The ability to have common references—sets of models and simulations—can play together and give you good information consistently. No model is perfect but some models are better than others. The ones that are repeatable and give you good fidelity are crucial. That is why this area is definitely a best practice.

In the software area in this distributive subsystem, another thing that was an excellent best practice was the early adoption of the systems software-engineering environment (S/SEE).[1] [That is] a big payoff when that happens because of the ability, in this distributive work force, to share in a proper way.

One of the key things when we do an independent technical course system is use of a consistent methodology. The paradigm is really like the Ballantine beer commercial—with the three rings, not the five. One [ring] is people, the second is process, and the third is product. (See Figure 1.)

From the people perspective we look at all of the people. What is the intellectual capital invested in creating an F-22? It is not just the primary contractor resource; it is what total resources the government brings to bear to create the system. What are the resource requirements of the government in conducting business as a better buyer vis á vis a better builder?

Figure 1. *The Independent Technical Assessment Methodology*



When you go to processes, it is not just the key process areas that are in the Capability Maturity Model®. It is all the methodologies, the procedures, and the tools linked together that gives these people, the willing workers, the wherewithal to create the product.

Finally [there is] the product. What is the corpus? What is its nature? What is its design? What are the attributes that have to be manifested for it to provide the full functionality that you will need? Underpinning all this is the ability to view operational requirements in the classic context. The program director [Maj. Gen. Michael Mushalla] is enlightened in this regard. He insists that everyone understand they are trying to produce the product with the needed operational functionality; it is the operational requirements, not just the specification, which is nothing more than trying to represent operational requirements technically. It is everyone knowing that the F-22 has to fly this fast, this high, at this degree of stealth, for this mission. That is the way this has been approached and we applaud it. It is one of the best we have seen in 30 or 40 years, and that includes the testers like Alpha Tech in the early involvement and at Langley [Va.] with its SMO22.

Even if you are the acquirer, maintain that perspective. Your job is not to satisfy a specification that has derived maybe logically, maybe illogically, from operational requirements. [Your job is] to produce the fighting machine. It is a different mindset. It is the enlightened leadership of the system program director that permeates the whole organization, both on the government side and the contractor side. Here is a best practice that has emerged out of that over the last decade.

You are creating a system that will exist beyond the middle of the new century. It is going to be here in 2050. Think of the F-15. The F-15 is not programmed out of the inventory until 2026, at which time it will have been in [the inventory] for well over 50 years. [This] creates a huge challenge.

The biggest challenge is not to get caught myopically in one program phase and begin making technical decisions or business decisions that do not consider the whole life cycle. Although they are very important, we do not do business reasons. We do not do politics. We do not do programmatics. We do *technical considerations.* We think there are plenty of other people who can do the political or programmatic aspects. It is true that occasionally business decisions or programmatics constrain a technical solution, sometimes to the degree where there is no remaining technical solution space.

"The biggest challenge is not to get caught myopically in one phase and begin making technical decisions or business decisions that are not considering the whole life cycle."

CROSSTALK: What is the underpinning of that enlightened leadership? Do you depend on a lot of research? How do you get to that stage?

Brandt: It is hard to quantify leadership. The big problem was not that they had not created a lot of intellectual capital, but to maintain that in our very fluid society.[2] You have a learning curve. The great danger is to lose momentum. There are two things that are the long-term risks, people and creeping obsolescence, as in the software-engineering environment. That has to be reinvested in and maintained because it is a needed tool from beginning to end. Also, you need simulation and modeling capability for the life of the system.

There is going to be evolution of software. Components are going to be evolving to avoid obsolescence. The long pole in the tent is not only managing the viability of the workforce but the tools, like the environment. More importantly we believe that diminishing manufacturing resources is the biggest long-term threat to this system.[3] Look at the evolution of our industrial base. What is happening beyond the millennium? The trend in the aerospace and defense industries has been one of blurring what those companies do. Many have become very large, like Lockheed Martin, for example.

What does it mean when we have vendors who are disappearing, going out of business? The challenge becomes managing the diminishing resources as vendors come and go, and the recognition that you are going to have continuous technology insertions. Do any of us have any insight into the hardware that will exist in the middle of the new century? Go back to 1950 and look at all this. There have always been prognosticators; on the scientific side they have tended to be conservative in general, and they have always poohpoohed the likes of Jules Verne or anybody who is the out-of-the-box kind of thinker because [those types] threaten the conventional paradigm of institutions.

CROSSTALK: How did you tie testing to requirements and to processes?

Brandt: You should test the requirements. You should always be geared to look at the operational performance of the system. You should not get sucked into worrying about the inner workings or hidden mechanisms. That is really the proper function for developmental testing. The ultimate in operational test and evaluation is going back and saying, 'What is this thing supposed to do? What are the operational specifications? Is it satisfying all these operational specifications?'

In our car, there is plenty of software. Do we care about that or do we care about the car's performance? We care if someone says the brakes went out because of the software. You need to do rigorous engineering during engineering and manufacturing development and development test and evaluation. But as you phase over to operational testing and evaluation your mantra has to be the requirements that were stated by the operator. You are delivering capability to him. You are delivering not just a piece of hardware but a weapon system that is conforming to the requirements that the Air Force has to fly and [use to] fight.

CROSSTALK: What made that F-22 testing successful?

Brandt: It is the early involvement and the ability to stabilize these requirements. Any requirement will evolve. There has always been a lot of negative thinking attached to the term 'requirements.' The plain fact is every system as it evolves has technology inserted, or as new missions emerge you discover more about the system's potential. It simply evolves. There is hardly anything anymore that would be

The Capability and Maturity Model and CMM are registered in the U.S. Patent and Trademark office to Carnegie Mellon University.

considered a static or set piece system.

The B-52 is going to have been in the inventory nearly 100 years. Does it look anything like the original avionics suites that were inside the airplane or even the engines? I don't think so. It has had this long century of evolution as a platform. The F-22, even though it is going to be the most capable fighter aircraft we have ever fielded, would be nothing more than good for air shows if it did not have this highly capable avionics suite where the heart of the aircraft resides and where the ability to deploy weapons is focused.

You must maintain capacity, capability, and full commitment of your workforce. It must be continuously managed —the keeping of the experience, the intellectual capital, all that you would need to prosecute the rest of this program because we will be tending this system for another 50 years.

Adopt that mindset when it comes to the evolution of hardware. That makes the issue of process, and institutionalizing good practice, so important because you know those who are here today are not going to be here 50 years from now. How do you carry forward the legacy of understanding? Sometimes you carry it through highly defined processes that have matured over time, and the accompanying documentation, so when you lose good ol' Charlie you do not lose a huge chunk of your memory and your experience, creating a void and therefore creating problems.

[Regarding Integrated Product Teams], we found them relatively mature across this program, beginning with Lockheed Martin.[4] As we went out and across the next tiers, they actually do exist. It is not the government over here pretending they are integrated, or the contractors, or the vendors. They are truly integrated and that is a great risk mitigation strategy because what it gains is better understanding overall of exactly the condition and state of the program. This intercommunication that goes through IPTs is quite mature.

There is a special kind of integrated product and process development, a Software Engineering Integration and Test Team. That is crucial when you come to this phase in a program [that is] highly complex; lots of things have to fit together, lots of contractors [are] involved, and now you are going to try to prove out this highly complex article.

CrossTalk: How do you handle discrepancies between testing and requirements?

Brandt: Sometimes it depends on when a discrepancy is known or not known, how you can or cannot recover. Often you have to do some negotiation and do that up front. The tester, if he is the operational tester, is constrained. I do not think you can negotiate a lot. [The tester] has the operational requirement; he is trying to validate that. What if there is some variance? There would have to be negotiation between the requirements persons—the combat command—and the development persons. It is always good to have a good understanding of what things are so sacrosanct that they could not change—the degree of stealthiness; performance parameters related to the aircraft; or more importantly, performance parameters that focus around the avionics and the functionalities you need to deliver weapons.

The evolution of complex systems, especially knowing that they are going to be legacies for a long time, is the proper mindset—nothing is totally static and all things become negotiable. They are negotiable in the sense you are resource-constrained, you are mission-constrained. There are various constraints that need to be looked at. You probably, over time, come to these suboptimal solutions that are at least within the solution space where you have a minimal satisfaction of all of these considerations.

CrossTalk: Which did you consider the greatest challenge—testing the hardware or the software?

Brandt: It is not an either/or situation because these are all software-intensive components. When you have embedded software, it is increasingly difficult to rationalize separation of pure hardware and software. You do not separate them. You make sure you have plenty of testing in the systems component area.

I have found those who have experience with software tend to make better systems engineers than those who came solely up the hardware route. I suspect that not all engineers agree with this finding.

Notes

1. See page 9 and 14 for more on the Independent Technical Assessment of the F-22 as it relates to the S/SEE.
2. See page 10 for comments by Ron Dubbs of Wright-Patterson AFB on engineer turnover.
3. See page 9 for Dubbs' comments on diminishing manufacturing resources.
4. See page 4 for Maj. Gen. Bolton's comments and page 9 for more on how the IPTs were found to be highly effective in the F-22 program.

F-22 Avionics Integration On Track

Boeing vice president and F-22 program manager Robert Barnes discusses how the company is reducing avionics risks and development costs by using both its ground-based lab and 757 Flying Test Bed to evaluate and troubleshoot the integrated avionics software long before it is installed on the F-22 Raptor. Barnes explains the software integration process and says that the F-22's avionics already have been through more rigorous testing than any previous fighter at a similar stage in its development.

by Robert Barnes
Boeing Vice President and F-22 Program Manager

Available online at www.stsc.hill.af.mil

A View from Wright-Patterson Air Force Base

CROSSTALK spoke with Ronald D. Dubbs, F-22 Weapon System Chief Engineer at Wright-Patterson Air Force Base, Ohio. He has technical responsibility for overseeing the development and production of the F-22 Advanced Tactical Fighter. He led the F-22 Avionics team for two years during Engineering and Manufacturing Development before assuming his current position. The following article is based on an interview with Dubbs and the draft of the Independent Technical Assessment of the F-22 Avionics written in October 1999.

According to the Independent Technical Assessment (ITA) accomplished by the Software Engineering Institute (SEI), the F-22 fighter aircraft is a Department of Defense (DoD) acquisition of unprecedented scale with a highly complex avionics suite.

Last year, a 10-member ITA Team from the SEI conducted an independent technical assessment of F-22 avionics.[1] The team was made up of a wide range of technical backgrounds, including computer science, systems engineering, avionics, space systems, F-15 simulation development, and integrated product and process development. The ITA Team found that major technical challenges for the F-22 included integrating the hardware and management of integrated resources, fusion of sensor data, and automation of pilot tasks. The major managerial challenges include the large number of contractors and the highly distributed work force.

The ITA Team produced findings in three separate areas:

1) People
2) Process
3) Product

The ITA found that using integrated product teams (IPTs) to develop F-22 avionics was highly effective and a good model for continued use.[2] They also found, with regard to people, that maintaining the capacity, capability, and full commitment of the entire work force is vital to the continued success of the F-22 program.

In the process area, the ITA Team found the systems/software engineering environment (S/SEE) is critical to the success of the F-22 avionics software,[3] common technical processes are defined and used across the avionics IPTs, and management processes anchored in the IPT approach are highly effective. (See Figure 1.)

Among the findings in the product area were that sensor data fusion has a significant impact on overall system effectiveness, and that there are known performance issues with the common integrated processor (CIP) for real-time avionics performance.[4] The team also learned that anomalies observed during testing of the inertial reference system (IRS) at Edwards AFB in the summer of 1999 had been resolved.

The ITA Team found that diminishing manufacturing sources (DMS) is a major issue affecting the F-22 over the weapon system's life cycle.[5]

"It is a problem you just cannot solve," agreed Ron Dubbs, Chief Engineer for the F-22 Weapon System Products (formerly the F-22 Avionics IPT Lead). "You can come up with a strategy to deal with it, but you cannot solve it," he said. "It definitely caused us to do some major redesign efforts to recover from that."

Decisions must be made on whether to make a lifetime buy of old technology or go to redesign, he said. The ITA Team learned that DMS has been identified by the System Program Director, Maj. Gen. Michael Mushala, as a major issue for the F-22. The ITA recommended adoption of a longer range horizon for looking at DMS in the F-22, as well as to a larger Pentagon Engineering Office Fighters/Bombers portfolio strategy.

In addition to DMS, Dubbs noted that keeping good people was not easy to do. The F-22 team had a significant turnover in software engineers, particularly in geographical areas with strong, competitive markets.[6] A steep learning curve and on-the-job training requires existing staff to help train new staff. The lengthy DoD security clearance process and difficulty in finding people with avionics domain experience further compounds the problem.

In other findings, the ITA team thought that the F-22 program has done a much better job than most avionics systems development programs in providing an architecture to guide designers and implementers.

"The architecture was set early on (1991-92), particularly the CIP architecture," Dubbs said. "That architecture went through the straw man, wooden man, and iron man models, before being definitized."

Dubbs added that the subsystem IPTs were provided with processor development stations early in their software development cycle, followed by partial CIPs, and full CIPs when needed. This allowed the IPTs to check out their software on the host system before sending it to the Avionics Integration Laboratory (AIL).

"Interface control has also been a challenge," Dubbs said. Due to the complexity and interaction of the avionics subsystems, establishing and controlling interfaces is very important. The F-22 program has developed and is using automated tools to track interfaces. Without automated tools, the management of interfaces for 108 computer software configuration items (CSCIs) with more than 2 million source lines of code would be overwhelming.

Asked how they tied testing to requirements, Dubbs replied that all F-22 functional requirements and test requirements from the weapons system specification down to lower-level component specifications are documented and tracked in an automated tool called Requirements and Traceability Management (RTM). Using RTM, it is possible to select a requirement and trace its associated links up and down the requirements tree. Links between requirements and test are also maintained in the RTM database.

Dubbs identified an extensive array of testing facilities that have been absolutely critical to the program success thus far. The primary facilities include the Vehicle Management System

Integration Facility (VIF), the Vehicle System Simulator (VSS), and the Tactical Aircraft System Integration Laboratory, all of which are located in Fort Worth, Texas. Also included are the AIL and the Boeing 757 flying test bed in Seattle and the Avionics Pole Model located at the Rome Air Development Center in upstate New York.

"We have the ability to test in open-air environments . . . [which is] a more dynamic form of test," Dubbs said. (See images below.) The Block Ø operational flight program (OFP) is successfully flying on aircraft 4001, 4002 and, most recently, on aircraft 4003 (delivered March 6, 2000). Integration, test and certification were performed in the VIF and the VSS. For the remaining software blocks (Blocks 1, 2/3S, and 3), the OFP will be certified at the Boeing AIL in Seattle and flown on F-22 test aircraft 4004 through 4009.

Dubbs noted Block Ø has been very successful to date and that Block 1 avionics testing on the F-22 will commence this summer at Edwards AFB, Calif.[7] (See Figure 2.) As noted by the ITA Team, the widespread use of modeling and simulation in the F-22 program is commendable. Modeling and simulation tools used for testing and analyzing mission software, in particular sensor track fusion, have been very effective.[8] The flying test bed provides a means to conduct real-time dynamic flight environment testing using real F-22 avionics sensors and laboratory instrumentation to evaluate integrated system performance, which helps reduce impact on F-22 flight schedules.

Reflecting on what they would do differently, Dubbs said, "We probably should have planned on more assets to keep those laboratories going. Adequate assets is a key thing."

If program funding allows, laboratory assets should remain in the lab and not be planned for transition to flight test aircraft. This type of situation creates an asset shortfall when particular laboratory efforts are extended due to schedule pressure. Another lessons learned is the common target processor and software runtime tools should be commercially available and mature by the time the application software is ready for integration.

"We should have gone more to common products and toolsets," Dubbs said.

The System Program Office/contractor team identified best practices that were agreed to and validated by the ITA Team. These included the use of a common S/SEE and common procedures across IPTs, the widespread use of modeling and simulation, and IPT implementation.

The ITA team noted that the S/SEE components are outdated and there is a growing risk that vendors will no longer support the common tools. The ITA Team was also concerned about the lack of validation of the simulations. Due to budget constraints those validations will not occur.

The ITA Team also thought that there is a risk to the continued success of the IPTs as some parts of the program wind down and experienced people move to other programs. This, however, is a normal occurrence in major development programs and program management must plan for personnel changes as the program moves to a maintenance mode. Minimal staffing for maintenance is the norm. Budget constraints that limit travel and face-to-face meetings may also adversely impact some IPTs. Additionally, the ITA Team recommended involving all stakeholders in any future revisions to F-22 Team Joint Procedures. The ITA Team expressed concern that management is focusing on budgetary issues and tracking known problems rather than using a set of techniques for identifying risks and implementing a mitigation strategy.

The F-22 Team and the SEI ITA Team identified the same best practices and agreed to a large degree on software risk reduction processes, while at the same time sharing many of the same concerns for life cycle risks.

The first avionics Raptor (4004) is scheduled for its first flight in the skies over Marietta, Ga. in early summer. Following several check out flights, aircraft 4004 will ferry to Edwards AFB to begin avionics flight testing. The flight-certified version of the Block 1 operational flight program was to deliver to the airplane in April 2000. The first flight of Raptor 04 will be another major milestone for the F-22 program.

Notes

1. See page 14 for more on the Independent Technical Assessment.
2. See pages 4 and 5 for Bolton's comments on the IPT as a best practice. See page 8 for Brandt's comments.
3. See page 6 for Brandt's comments on the S/SEE.
4 See pages 13 and 14 for comments by Beverly Moody of Wright-Patterson AFB on the CIP.
5. See page 7 for Brandt's comments on diminishing manufacturing resources.
6. See pages 7 and 8 for Brandt's comments on software engineer turnover.
7. See page 4 for Bolton's comments on the block approach.
8. See page 6 for Brandt's comments on modeling and simulation.

Fort Worth

Avionics Pole Model

Seattle

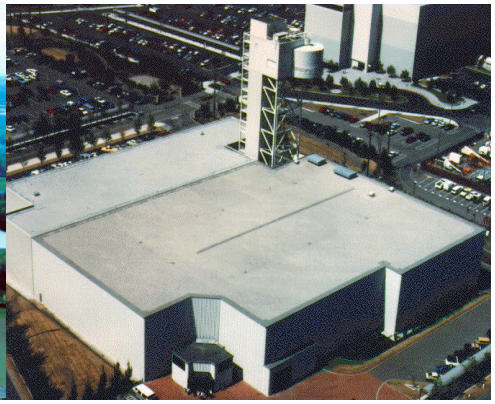


Figure 1. *F-22 Team-Wide Software Development Processes*

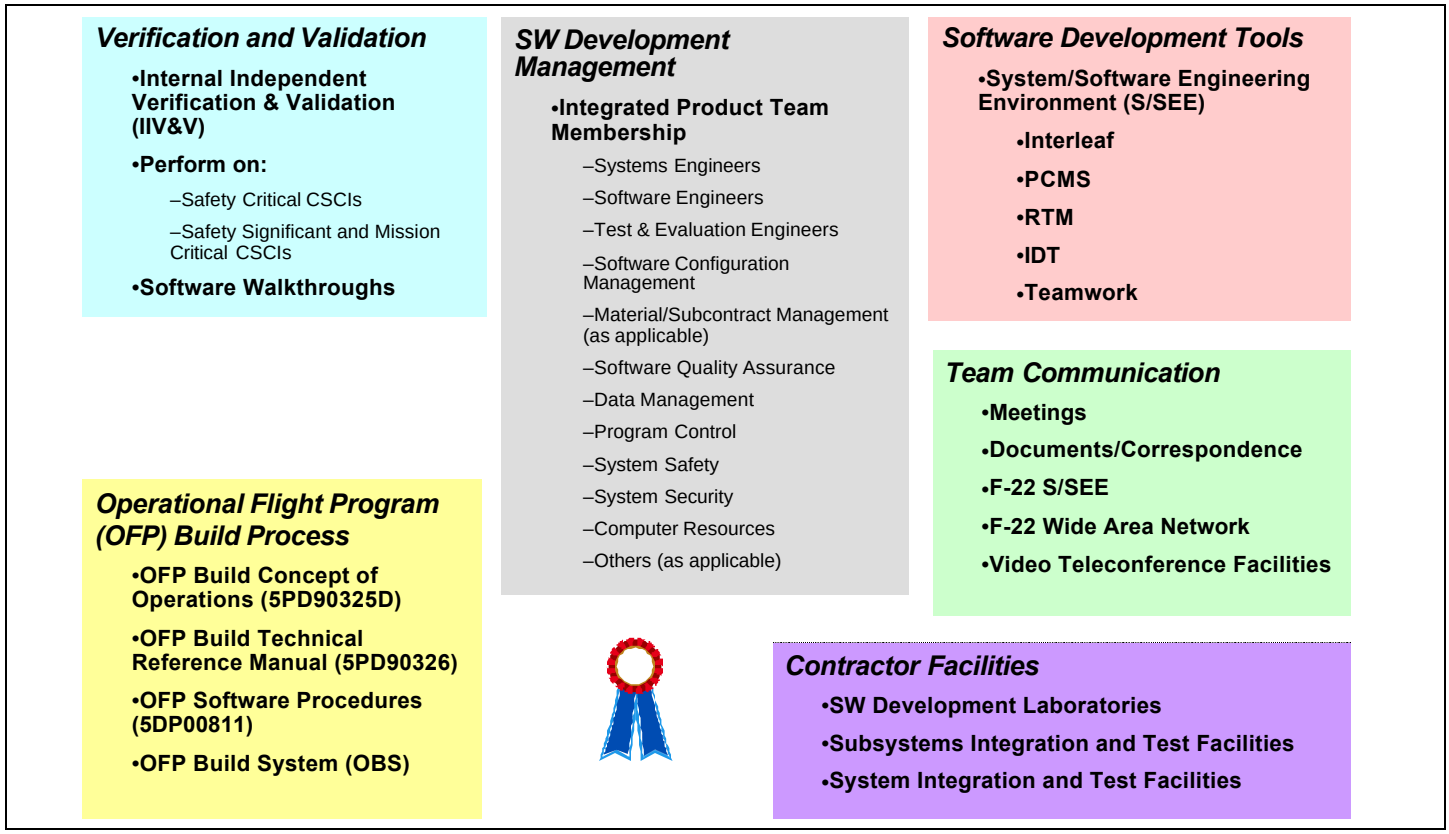
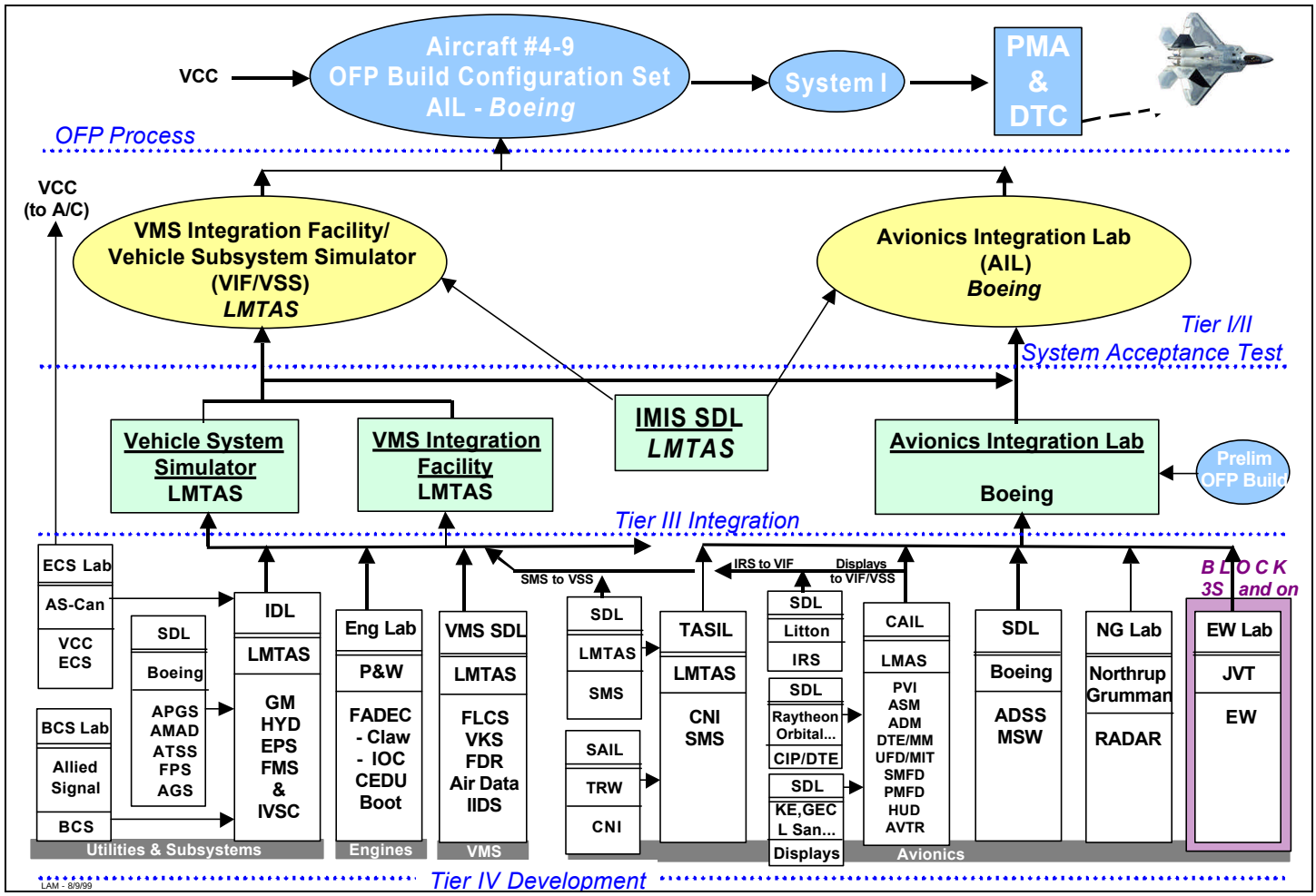


Figure 2. *F-22 Block 1-and-On Integrated Systems Product Flow*



F-22 Software Risk Reduction

by Beverly L. Moody
F-22 Avionics Software Block Lead

In the early phases of the Advanced Tactical Fighter (ATF) program, the Software Program Office (SPO)/Contractor Teams realized that serious efforts would need to be applied to mitigate the risk associated with developing well over a million source lines of code (SLOC) across many development sites throughout the United States, Canada, and Europe. The ATF software development team was composed of the three prime contractors, four major suppliers, and more than a dozen smaller suppliers, all of whom had to have a common goal of developing an operational flight program (OFP) for the ATF. Immediately after the contract was awarded in 1991, the Computer Resource Working Group (CRWG) and the Systems/Software Engineering Environment (S/SEE) Integrated Product Team (IPT) set out to build the framework in which the F-22 OFP would be built.

It was almost 20 years ago when the Air Force identified the need to replace the aging F-15C air superiority fighter. In 1986, two teams received the ATF Demonstration/Validation (Dem/Val) contract: Northrop Grumman and the eventual winning team composed of Lockheed, Boeing, and General Dynamics. During the Dem/Val phase, areas of risk were identified and plans to mitigate those risks were implemented. Not surprisingly, software development risk mitigation was near the top of the list, since the ATF was to be the most software-intensive airplane built for the Air Force. The ATF Team planned to develop approximately 1.5 million source lines of code, across more than 20 software development companies located throughout the U.S., Canada, and Europe. The main focus for software risk reduction during Dem/Val was the need for a common development environment and team networks to enable rapid reliable communication across the various sites. This article focuses on software risk reduction practices in three risk areas associated with F-22 software development:

1. Development schedule.
2. Complexity of the development team.
3. Leading-edge technology challenges.

F-22 Development Schedule

The Engineering and Manufacturing Development (EMD) contract was awarded to the Lockheed/Boeing/General Dynamics Team in August 1991. The F-22 Raptor's first flight was six years later in September 1997. Initial flight of the first avionics airplane is scheduled for late this spring, and Raptors will enter the operational environment in 2005. The time span from identified need to operational capability is almost 25 years.

The technological advances in the areas of computer hardware, software, and software development processes have been, and are expected to be, phenomenal. Keeping pace with these rapid advances over a protracted development schedule has been a continuing challenge. Major changes in the acquisition process, particularly the move from the DoD-mandated military standards to reliance on commercial standards and industry best practices, generally facilitated, but at times exacerbated, risk mitigation as contractor and government define their new roles.

Software Development Environment

Based on experience and lessons learned from earlier programs, the Lockheed-led team wrote specifications for the tools that it thought needed to be included in the software development environment. Digital Equipment Corporation (DEC) was selected as the provider of the host environment that was VAX/VMS based. DEC was tasked to identify a set of tools that would meet the F-22 team's specifications, and to present the entire environment as a turnkey package to the team for approval.

Requirements for an interface definition and control tool were included in the specification. These were the only requirements provided to DEC for which DEC could not find a commercially available tool, so DEC committed itself to developing the Interface Definition Tool (IDT).

DEC selected Interleaf as the document publication tool, Teamwork for requirements analysis, Requirements and Traceability Manager (RTM) for requirements traceability, and Product Configuration Management System (PCMS) for configuration control of software, documents, and software development files. The common systems/software engineering environment (S/SEE) for all F-22 software development was identified and installed at multiple contractor/government sites by early 1992.[1]

In 1991, VAX/VMS was the only environment which met the team's technical requirements as well as the F-22 program security requirements for wide area networked computer systems.

Even back in the early 1990s, when we were not operating at Internet time, the team recognized that VAX/VMS days were numbered, and that the F-22 would outlive this mainstay computer system. For long-term risk reduction, the F-22 Team worked with DEC to identify a migration path that would sustain the S/SEE capability. This migration path led to a yet-to-be built environment called Cohesion that was to be based on the Alpha workstation with open VMS or UNIX. DEC never managed to bring this together in an affordable and functional package for the F-22 Team. DEC no longer exists, and Compaq, which bought them out, no longer takes orders for, or offers long-term support for, VAX computer systems. The F-22 team continues studying S/SEE migration for the near future, and will have to continuously plan to mitigate the risk of changing environments over the F-22 life cycle.

The common S/SEE and the associated networks have been invaluable to F-22 software development. The networks link the contractors, System Program Office, Air Combat Command, Air Force Operational Test and Evaluation Center, and Edwards AFB Combined Test Facility at the appropriate security levels. This connectivity provides the SPO and other government offices with nonintrusive insight as we can access common areas and desktops for documents, briefings, metrics, and software schedules.

There will always be a need for some commonality among the software teams for populating and disseminating information from the air vehicle-wide databases such as IDT and RTM. The IDT database is an integral part of building the operational

flight program as it supports the trusted computing base on the Raptor. Commonality at the software code development level may not be as critical today as it was 10 years ago due to the intercommunication between computer systems, which is a lot easier than it used to be.

Out-of-Production Processors

The old way of doing business—freeze the processor, tools, and compiler at some point, and stick with it, is no longer practical in today's high-paced developments. Processors are becoming obsolete at a rapid pace, and frequently new compilers and tool sets are not developed for old systems. Because compilers and tool sets for the new processors, which are replacing out-of-production processors, are not available in the VAX/VMS environment, the F-22 team is already using other host environments for some code development. Military developments will have to migrate processors and environments at the commercial pace or lose some of the benefits of commercial off-the-shelf products.

Commercial Tools and Security Issues

One other lesson learned from dealing with commercial tools is that commercial tools do not inherently deal well with classified information. For example, rolling up paragraph markings to the correct markings at the top and bottom of each page is not easily done by most commercial tools. Adding a new paragraph to a baselined document may take some major editing. Coordinating/contracting with tool developers to provide some program specific features dealing with classified markings may be necessary for future programs as well as the F-22.

MIL-STD-2167A Mandate

At the time of F-22 contract award, software development fell under the Ada mandate and MIL-STD-2167A. MIL-STD-2167A was a standard to provide a structured process for developing software, testing the software, and documenting same. Under DoD Acquisition Reform policy, most standards were deleted and the Ada mandate was eliminated. The F-22 program deleted most of the contract data requirements list in 1996, however, most of the F-22 team software developers continue to produce the MIL-STD-2167A documentation and have it available on the S/SEE. The commercial standards that replace MIL-STD-498 and MIL-STD-2167A are based on the military standards for software development, since those standards were the best available.

Ada Mandate

Ada 83 will continue to be the primary language used on the F-22 (80-85 percent) for the foreseeable future since much of the code is already complete. Some teams are looking at migrating their Ada 83 code to Ada 95 since most of the new compilers are based on Ada 95. To date, needed source code changes have been trivial and the migrations fairly straightforward. The promise of the portability of Ada has been validated by several programs.

Complexity of the Software Development Team

The completed F-22 Air Vehicle OFP will contain more than 2.2 million SLOCs. There has been some growth as well as added functionality, which have added to the original estimate of 1.5 million SLOCs. Our software developers include the three primes, several major suppliers, as well as smaller single function-type developers. Some companies are developing less than 5000 SLOCs; one major supplier is developing more than 500,000 SLOCs.

Teammates and Competitors

Some of the F-22 teammates compete against each other for other program contracts. In a highly integrated weapons system development, regular communication and coordination between designers and developers is essential. This was a new and unusual environment for most team members. The F-22 team has overcome these obstacles, which is a tribute to the various corporate management teams that have allowed their software developers to work directly with competitors in a major cooperative effort.

Ways of doing business within a company, the corporate culture, are frequently very different, and compromise has become a way of life for the F-22 team. For example, each company had its own change notice forms. Some companies used interface revision notices; others used interface change requests. Terminology and forms for a highly integrated development required F-22 specific terms and forms to aid in communication.

Building the F-22 Team

Some of the risk-reduction efforts undertaken to bring this diverse and geographically separated team together to perform a major highly integrated software development are listed here. Lockheed /Boeing/General Dynamics selected the Software Productivity Consortium's Ada-based Design Approach for Real-Time Systems as the team software design methodology. The S/SEE Help Desk located at Lockheed in Marietta, Ga. had to support multiple time zones from Rochester, England to the U.S. West Coast. As the teams started to use the S/SEE with all the associated development tools, there were many cries for help.

Working groups were established with representatives from the primes and major suppliers to determine the processes and procedures for using the various tools to ensure a common approach was used across all the developing sites. Training in the use of the S/SEE tools was provided at various locations throughout the team.

The F-22 Team

One outcome of all these efforts is that the team comes together (via meetings, telecons, video teleconferences, etc.) speaking a common language. Some team members share Ada package specs for interface definitions. Developers of applications that run in the Common Integrated Processor (CIP) come together in a weekly telecon with the CIP Help Desk to work through issues, problems with CIP tools such as debuggers, share workarounds, compiler experiences, and to ask for advice from other users to resolve problems or concerns.[2]

Team-Wide Problem Reporting System

Delivering a product with a known problem at the developing site but not passing the data along to the receiving site, can lead to a lot of wasted time as the problem is rediscovered. With many developing sites, accurate and complete problem report tracking is essential. The Team's Common Problem Reporting System (CPRS) was implemented to ensure complete tracking of all air vehicle problems. Problem reports on delivered products are maintained in a single master database at Lockheed. Problems are categorized by severity, and move through various states until the initiator verifies them as closed. CPR Boards meet via telecon

or VTC on a regular basis to disposition problems, determine when the fix is required, and assign the fix to a block and a particular build. Developers, labs, manufacturing, and the test pilots have access to the database, the opportunity to participate in the Boards, and have an input to the dispositioning of problem reports. Problem report closure is one of the critical metrics used to determine the readiness of a new build.

F-22 Leading Edge Technology Challenges

Integrated Avionics

The most challenging technical requirement for the F-22 avionics software is to successfully achieve sensor fusion. Sensor fusion combines multiple sensor target attribute data into one target track file for presentation to the pilot. Shortcomings of any one sensor will be overcome by fusion of all sensor data. Fusion has not been done to this scale on any other tactical military aircraft platform. The mission software team uses simulations and an Algorithm Prototyping and Analysis Tool to reduce this development risk.

Lessons learned from other programs told the F-22 team that it was quite common to use up the memory and throughput of the mission computer with the first delivered OFP. The CIP architecture was designed and sized for growth in functionality. Two CIPs are included in the current design with growth for a third for a minimum 300 percent long-term growth with technology insertion. There are 66 slots available in each CIP. CIP 1 has 19 slots open and available, and CIP 2 has 22 slots open and available. If necessary, but not in the current plan, more processors could be added during the EMD timeframe if the reserve required in each processor gets used up. Memory and throughput are monitored at the subsystem level on a regular basis.

Incremental Integration

Raptor 01 OFP integration occurred in integration labs at Lockheed, Fort Worth. OFP integration included flight controls, utilities and subsystems, displays, inertial reference system, and stores management system. This OFP included more than 700K SLOCs, is flying at Edwards AFB on Raptors 01 and 02, and has nearly 600 hours of flight test. Raptor 03 had its first flight in March 2000.

Integration risk is mitigated by incremental levels of integration. Subsystem (radar, electronic warfare, communication, navigation and identification, etc.) hardware and software integration is initially accomplished at the subsystem level by the developing Integrated Product Team (IPT).[3] Software is sent to an intermediate lab or to the Avionics Integration Laboratory (AIL) at Boeing.[1] Partial blocks of software are sent to the Flying Test Bed (FTB), a modified Boeing 757 with a wing and sensor edges in addition to some of the avionics hardware, for further testing in an airborne environment.[4] The AIL has responsibility for the integration and certification of the Air Vehicle OFP for Raptor 04 through Raptor 09, as well as the production representative air vehicles and Lot 1 (Raptors 10 -27).

Summary

Metrics is one method used to measure success in our risk mitigation efforts. Some of our earliest metrics tracked the number of calls to the S/SEE Help Desk by tool and severity (from user inexperience to immediate change needed to the tool). SLOCs were tracked, along with notes explaining any significant jumps. Subsystem metrics were kept on code and unit test, computer software component integration, and formal qualification testing. Problem report metrics by subsystem as well as metrics on AIL tasks, functions, and system acceptance test procedure completions are tracked on a weekly basis. Many metrics have changed as we moved through various phases of the software development cycle to meet the current need.

Team communication paths, whether on telecons, video teleconferencing, the S/SEE, or the PC, have played a major role in F-22 software development. Problem areas are identified early and given needed attention to reach resolution.

The Software Engineering Institute (SEI) was hired by Maj. Gen. Claude Bolton, Air Force Program Executive Officer for Fighters and Bombers, to perform an Independent Technical Assessment (ITA).[5] Their findings were very positive in the area of software risk reduction efforts undertaken by the F-22 Team. Out-of-production parts and diminishing manufacturing sources were identified as one of our biggest risks.

The next few years will be exciting and challenging times for the F-22 program. We will finish building the EMD planes, continue with flight test, move into Initial Operational Test and Evaluation, start production, and prepare to go operational in 2005. The F-22 Team expects to find that software risk mitigation efforts pay off.

Notes

1. See page 6 for Thomas Brandt's comments on the S/SEE and page 9 for Ron Dubbs' comments.
2. See page 9 for comments on the CIP by Ron Dubbs' of Wright-Patterson.
3. See pages 4 and 5 for Maj. Gen. Calude M. Bolton's comments on the IPT as a best practice See page 8 for comments by Maj. Gen. (Ret.) Thomas Brandt. See page 9 for Dubbs' comments.
4. See page 4 for more on using the flying test bed in the F-22 program.
5. See page 9 for more on the Independent Technical Assessment.

About the Author

Beverly L. Moody is the Avionics Software Block Lead for the F-22 Avionics Integrated Product Team, F-22 System Program Office, Aeronautical Systems Center, AFMC, Wright-Patterson Air Force Base. Her responsibilities include managing incremental software block deliveries to the Avionics EMD aircraft, Raptor 04 through 09. Moody began her Air Force career in 1982 at the Language Control Facility working with JOVIAL and Ada standards and compiler validations. She worked for SEAFAC performing MIL-STD-1750A computer validations and Ada training. She was then assigned to the tri-service Mark XV System Program Office as the computer resource lead working on a planned replacement to the Mark XII Identification Friend from Foe system prior to her current assignment to the F-22 SPO in 1991.

ASC/YFAAT
2130 Fifth Street
Wright-Patterson AFB, Ohio 45433-6503
Voice: 937-255-7503, Ext. 2457
Fax: 937-255-1144
E-mail: Beverly.Moody@ASC-YF.WPAFB.AF.MIL

Letter to the Editor

Dear CROSSTALK:

Keep up the excellent quality—by far the best magazine of its kind for quick and accurate information, and the latest in software development info.

Thank you,

Jim Syrris
McKessonHBOC

Coming Events

May 22-23
6th Annual Montgomery Golf Outing and Information Technology Partnership Day
http://web1.ssg.gunter.af.mil/partnership

May 30-June 2
13th International Software Quality and Internet Quality Week (QW 2000)
www.soft.com/QualWeek/QW2K/index.html

June 4-11
22nd International Conference on Software Engineering
www.ul.ie/~icse2000

June 4-7
9th Biennial IEEE
http://cefc2k.aln.fiu.edu

June 5-7
2000 IEEE International Interconnect Technology Conference
www.his.com/~iitc

June 10-14
ISCA2000: 27th International Symposium on Computer Architecture
www.cs.rochester.edu/meetings/ICSA2K

June 18-22
ICC 2000—IEEE International Conference on Communications
www.icc00.org/

July 11-13
5th Annual Conference on Innovations and Technology in Computer Science Education
www.cs.helsinki.fi/events/iticse

July 16-18
7th IEEE Workshop on Computers in Power Electronics
www.conted.vt.edu/compel.htm

July 16-19
Congress on Evolutionary Computation
http://pcgipseca.cee.hw.ac.uk/cec2000

August 6-11
6th Annual Conference on Mobile Computing and Networking
www.research.telcordia.com/mobicom2000

August 7-8
IEEE Workshop on Memory Technology Design and Testing
http://pcgipseca.cee.hw.ac.uk/cec2000

August 17-19
Designing Interactive Systems (DIS) 2000

September 10-12
Collaborative Virtual Environments (CVE) 2000

September 10-14
Very Large Databases 2000

Visit www.acm.org/events for information on this, DIS, & CVE.

April 29-May 3, 2001
STC 2001: The Premiere DoD Software Technology Conference
www.stc-online.org

F-22 Quiz *by John Higgins*

1. When was the F-22 first flown?

Are you stumped?
Tempted to peek at the answers?
All of the information you need to ace this quiz may be found at
http://johnsf22.cjb.net

__January 3, 1998
__July 25, 1997
__September 7, 1997
__October 7, 1997

2. Who is the maker of the F-22's landing gear?

__Menasco
__Arcat
__Boeing support
__Lytic Engineering

3. The F-22 has what type of thrust direction?

__3-D thrust vectoring
__2-D thrust vectoring
__No vectoring
__High direction thrust

4. What type of power plant does the F-22 use?

__Lockheed Martin Inc.
__Pratt & Whitney
__Grummen
__Lucent Technologies

5. How many rounds of ammunition can the F-22 accommodate?

__745
__156
__250
__480

6. How wide is the F-22 wing span?

__66 feet 10 inches
__30 feet 2 inches
__44 feet 6 inches
__65 feet 4 inches

7. The F-22's radar is:

__Open Element
__Circular Element
__Active Element
__Big honkin' radar

Answers to Quiz

1. September 7, 1997
2. Menasco
3. 2-D thrust vectoring
4. Pratt & Whitney
5. 480
6. 44 feet 6 inches
7. Active Element

F-22 Web Sites

F-22 Raptor Stealth Fighter
www.f-22raptor.com/

This is a Boeing site that includes information on how the F-22 program was saved, news from Capitol Hill related to that program, and allows visitors to download the F-22 Raptor screensaver. It also links to the Lockheed Martin corporate Web site (Lockheed is the maker of the real F-22); Boeing corporate Web site (Boeing co-developed the current F-22 Stealth Fighter); and the Pratt & Whitney corporate Web site (they made the F-22 turbine engine).

YF-22 Lightning/F-22 Raptor
http://members.tripod.com/~F22FighterJet/

This links to pictures of the F-22 Raptor, the YF-22 Lightning 2, and specifications.

Raptor Rapture
www.pratt-whitney.com/features/raptor.html

This is an overview of the Raptor, including a description of Pratt & Whitney's F119 engine.

About the F-22 Raptor
www.geocities.com/TimesSquare/Ring/2960/about/f-22.html

This site gives some background on the F-22 (reduced observables, supersonic persistence, air-to-surface capability, etc.)

F-22: Creativity and Innovation
http://eagle.westnet.gr/~access/newpage6.htm

More information, with photos and a link to Boeing.

Fort Worth: F-22 Fighter Programs
www.lmtas.com/FighterPrograms/index.html

Lockheed Martin site includes links to press releases, photos, and mission briefs.

Stealth Fighters Web Site
www.geocities.com/CapeCanaveral/Lab/8004/index.html

Compare the F-22 to other stealth fighters.

Pictures and Technical Diagrams
www.aw.fl.net.au/ef2000/pictures.html

Great source for images of the F-22 Raptor and Eurofighter 2000.

Reducing Software Project Productivity Risk

by Richard Bechtold
Abridge Technology

Software project results continue to be highly subject to the skills of the individuals involved in the development life cycle. Although improved processes and tools can help increase overall productivity, it remains true that the project team is one of the greatest variables that impact project productivity. To significantly reduce software project productivity risk, it is important to simultaneously evaluate the processes, tools, and skills that characterize the project and its personnel. This article presents three approaches to reducing software project productivity risk by minimizing process rules, by maximizing simple tools, and by de-emphasizing the importance of technical skills in favor of basic abilities.

Rules, Tools, and People

Software-intensive systems development still persists as an extremely people-intensive effort that is highly subject to productivity variances. Software programmer productivity differences have been reported as high as 100-to-1 [1], and more recently as 22-to-1 [2]. Even worse, on a software project with 10 people, you can expect to have up to three developers who are "net negative producing programmers" [3]. These are people whose high rate of defect insertion more than negates their rate of code production. In effect, overall productivity on the project accelerates when such people do not show up for work. Clearly, enhancing programming productivity is essential for ensuring software project success. However, since an average of 84 percent of all software projects fail [4], it is also clear that we are not adequately addressing the problem of reducing software project productivity risk.

Three key components to reducing software project productivity risk are rules, tools, and people. Initially, projects often strive to make individuals more productive. This can be a mistake. Ultimately, what you need to achieve is to make the project, as a whole, more productive. This is accomplished first by thinking of the project personnel as a cohesive team, and second by providing this team with productivity-enhancing tools and rules.

The notion of productivity-enhancing tools is self-evident. However, what are productivity-enhancing rules? Simply stated, these rules are the processes in use on your project. Their purpose is to help ensure the project achieves operational or business objectives. Ironically, in striving to improve the processes, a misunderstanding of the differences between rules and tools can easily lead to bloated processes that reduce overall project productivity. If this is accompanied by the far too common tendency to staff a software project with the wrong types of people, then low productivity and high project failure rates are hardly a surprise.

This paper discusses how to improve overall software project productivity by avoiding bloated process descriptions, by leveraging simple tools, and by recognizing the team-member skills that best ensure the overall success of your project.

Leveraging Rules

Most process quality frameworks, such as the Software Capability Maturity Model (CMM®) [5] and the ISO 9000 set of standards [6], require the use of defined processes. The premise is that well-defined and institutionalized processes are a key component in ensuring the performance of efficient, effective, and repeatable activities. These high-quality activities, in turn, help ensure the production of high-quality products.

However, numerous projects and organizations have suffered through the agony of attempting to implement processes that were voluminous and hard to interpret. In particular, when a project manager is handed a 100-plus page process description, how does he or she distinguish between actual process requirements and the associated guidance, suggestions, or supporting material? Where is the project manager allowed to exercise discretion and judgement? As organizations encounter these problems, and strive to answer these questions, they often resort to attempting to develop process-tailoring guidelines. Occasionally, this effort can result in a set of process descriptions that are far more complicated, and correspondingly more difficult to use.

For example, if you have a notion of four types of projects (e.g., database-intensive, object-intensive, web-intensive, hardware-intensive) and four sizes of projects (e.g., small, medium, large, very large) you now have up to 16 different process variations to define and maintain. Add another dimension, such as customer type (e.g., Department of Defense, other government, industry, mass market), and you may be looking at up to 64 process variations.

Potentially the greatest factor contributing to these problems, and to the corresponding reduction in project productivity, is the difficulty process users have in discerning between those parts of the process descriptions that must be followed (the rules) and those that simply help with understanding and following the rules (the tools). A simple solution to this problem is to stop thinking about process in the abstract, and instead distill it to the component parts of either rules or tools. This will contribute to far smaller process descriptions that document only the essential elements, or rules, that must be followed. All other material is relocated to the tool side of the process.

As with standard process descriptions, there will typically be several layers within a process rule description. For example, at the highest level there may be a rule similar to the following:

- Projects will manage software configurations.

Subordinate to that rule, you might have the following five rules:

- Projects will plan configuration management activities.
- Configuration items will be uniquely identified.

- Changes to configuration items will be controlled.
- Configurations will be audited.
- Configuration status will be reported.

Each of these rules may have its own set of subordinate rules, and so on. Taking this approach can dramatically reduce the size of a process description by reducing it to its essential requirements. This results in a process representation that clearly communicates what has to occur on the project.

This does not eliminate the need for tailoring the rules to specific project characteristics and circumstances. However, there is far less material to tailor, and consequently the tailoring is far easier. For example, tailoring might simply consist of waiver criteria that define when a project is exempt from a particular rule. Implicitly, being waived from a higher-level rule results in a waiver from all its subordinate rules.

Additionally, the principle of moving all extraneous material out of the process descriptions provides the opportunity to apply a litmus test that anyone can use when developing, reviewing, evaluating, implementing, or discussing the contents of process descriptions. The test is to ask whether or not the item under examination is absolutely a requirement for the project. If not, then it does not belong with the process rules. It belongs with the tools.

Leveraging Tools

When we think of software project tools that enhance overall project productivity, we usually think in terms of automated tools that assist with project planning, project tracking, configuration management, action-item tracking, requirements management, systems design, regression test support, quality tracking, etc. However, as implied by the preceding discussion on process rules, anything that is not a rule but helps us interpret or comply with the rules can be considered a process tool. A few common, simple examples of such nonautomated tools include:

- Checklists
- Guidelines
- Templates
- Outlines
- Examples
- Training material

By putting as much process-related information and content into the above and other types of simple process tools, you reallocate your process information more closely to the ratio project personnel generally prefer. Most people prefer as few rules as possible. Conversely, they also prefer having a large assortment of tools available such as templates, examples, and checklists.

Another advantage to minimizing rules, and maximizing these types of simple tools, is the flexibility you have to regularly and rapidly revise, upgrade, augment, or otherwise improve the information, material, and workflow sequences associated with the set of tools that supports the project. Typically, if you want to change the process rules, your proposed changes will require a sequence of reviews, revisions, and approvals before they are authorized for use on the project. Conversely, in most organizations authority to change the miscellaneous material used to support the project is delegated to lower levels, and might even be within the authority of individual projects.

In addition to the simple, nonautomated tools presented above, you can certainly increase productivity through the careful use of automated tools. However, be cautious that the tools you deploy on the project result in an increase in the ability of the project personnel to perform their work. For example, you will usually want to avoid any automated tools that have long, steep, learning curves. Likewise, avoid tools that encourage excessively complicated approaches or solutions, since it is highly likely you will end up with precisely that. Conversely, tools that are easy to learn and use, and that encourage simple solutions to complex problems, can help significantly improve overall project productivity.

It is very important to clearly communicate to all project personnel that the overall objective of tool usage is primarily to improve project productivity. When a tool seems to interfere with productivity, its value must be carefully re-evaluated. Certain tools, such as a checklist that helps developers determine if their code is compliant with software coding standards, may initially seem to reduce productivity. It will take additional time for programmers to complete the checklist as they review their code for compliance to the standard. However, this is a good example of how something that might not seem productive for an individual in the short-term can be productive for the project over the long term. In this case, the checklist can result in higher levels of standards compliance, which can lead to increased readability, maintainability, and reusability of the software components. Each of these benefits can directly and significantly contribute to an overall increase in project productivity.

Although rules and tools are vitally important to software project success, their importance is dwarfed by the need to have the right type of team, consisting of people with the right types of skills. No combination of tools, nor cleverly crafted rules, can save a project that is chronically staffed with the wrong people.

Leveraging People

As described in the introduction, two software developers with similar education, skills, and salary can have a productivity difference of 22-to-1, or even greater. Put differently, Mary can get the job done in two weeks, but John will take nearly a year to accomplish the same result. Rules and tools cannot eliminate this extreme productivity variance. Further reduction of the risks associated with software project productivity clearly requires identifying and staffing the project with a highly productive team. But what traits allow a software team to be productive while pushing toward a common goal? Most organizations talk about finding the best people. But what do we look for when looking for the best?

Although it can be difficult to detail who the best people are, it is very easy to describe who the best people are not. Pick up the Sunday classified ads from your newspaper. Read the job descriptions for the information- and technology-related jobs. There you have it. A perfect prescription for how to identify the worst people. Using the Washington Post Web site [7], most technical, software project related job postings are looking for things like those shown in Figure 1.

Desperate ads look for six months of tool-specific experience. Choosy ads prefer two years. Occasionally, someone wants applicants to have a college degree. It is rare that an ad requires someone to be flexible, multitalented, or a team player. This is in spite of the fact that these three skills can be crucial to the success of a software project.

The following list provides examples of three levels of skills that can apply to any given software development project. The various skill areas are divided into the following levels: essential, important, and useful.[1]

Essential

- Able to learn quickly
- Willing to learn quickly
- Team player
- Generalist, or multiarea specialist
- Strong on fundamentals
- Very flexible
- Able to teach efficiently/effectively

Important

- Insightful
- Adept with abstractions
- Organized
- Disciplined
- Self-motivated
- Good negotiator
- Good problem-solver

Useful

- Strong understanding of the solution engineering techniques
- Strong understanding of the solution engineering tools
- Moderate familiarity with the underlying environment
- Moderate understanding of the problem domain (including under standing the customer/market)
- Familiarity with primary applicable standards
- Good estimation skills
- Good reporting skills
- Good leadership skills

Figure 1. *Breakdown of IT-related job postings*

Keyword	# of Listings
C++	3946
Java	1539
Access	2193
Oracle	2083
Visual Basic	892

The relative importance of various skills from the preceding list runs almost completely at odds with the typical approach most people use for staffing a software project. Typically, the selection criteria for staffing a project is based upon someone having a few years of experience in a particular tool. The above approach demotes tool-experience to the least important skill category. Instead, team skills, fundamental skills, interpersonal skills, and the ability to learn and change rapidly are given much higher value.

The rationale for reducing the importance of tool-specific technical skills and increasing the importance of fundamental abilities and character traits is that if software projects are characterized by anything, most are characterized by rapid changes in requirements, technologies, competing products, and opportunities. Given this environment, software development projects need to be staffed by people who are extremely agile and quick to learn.

Another key factor for determining how to assign relative importance to various skills is the general ease with which the lack of that skill can be fixed by the software project manager. For example, consider the useful skills. Typically, if a new developer does not know your customer, that is fairly easy to fix. Likewise, if the developer does not understand one or more of the tools you use as part of building the product, this, too, can usually be fixed rather easily. This is especially true if, as discussed earlier, the tools you use on the project are easy to learn, understand, apply, and yield simple and elegant solutions.

Similarly, consider the essential skills. It is very hard to teach someone to be a team player. They either naturally are, or naturally are not. Likewise, it is nearly impossible to change someone from being an inflexible person to a flexible person. Additionally, teaching someone everything they need to know to be strong on computer science and software engineering fundamentals is similar to trying to teach them the equivalent of a software technology-intensive bachelor's degree. While not impossible, this type of comprehensive and fundamental education is likely best left to academic institutions.

Skills at using the latest language, tools, and libraries are certainly important, but these sometimes are the most volatile and short-lived skills needed on a project. The rapid and sometimes extreme rate of change in tool capability, requirements, and opportunities may require you to reconsider the tools you are using on your current project, or cause you to prefer other tools for your next project. Having a flexible team with the ability to rapidly learn new tools seems preferable to trying to build a new team.

From a different perspective, the years of experience you seek from a job candidate may not equate to years of applicable experience for your project. For example, you may be better off with someone who has four months of direct experience on your project—even as a trainee—than someone who has a year of experience acquired somewhere else. Even the person with a year's experience might require a month or two to come up to speed in your environment. From this perspective, it can be better to quickly find and hire someone who only lacks your specific tool experience, than to let the position go unfilled even for a month.

In the contracting environment, much of the focus on specific tool experience is the result of standard, if dated, government procurement habits. Most procurements force a bidder to, for example, provide someone with two years of Common Business-Oriented Language experience. These are typically "must comply" requirements and leave the bidder with virtually no choice in identifying and providing the best overall team for a particular contract.

Without question, software development teams need to have at least some people who are adept at the various tools the project will use. No project can afford to be entirely a learning experience. Nevertheless, if most team members exhibit the essential and important skills described in this paper, and at least some team members exhibit the useful skills, then you have an excellent pool of available skills regardless of the circumstances the project eventually encounters.

Summary

It is hoped that this article has illustrated the potential value of minimizing the number of rules your defined process employs, and maximizing the number of

simple tools that support process rules.

But when it comes to further reducing software project productivity risk for a specific project by deliberately seeking personnel with certain skills, assessing the relative value of those skills remains a challenging task. The varying contribution that different skills make toward reducing project productivity risk truly does depend on your project's context, constraints and circumstances.

Nevertheless, here is a test scenario to consider:

You have been tasked to manage a complex software development project that spans several years, has uncertain requirements, and might involve uncertain technologies. A similar project recently finished—two years late—and it was built using power–wizzy–gui–whatever 1.5.

You must select a software development team from the two teams that are available.

One team has considerable experience using power–wizzy–gui–whatever 1.5, but it must comply with a 300-page process description. The team has few routine tools available; it is inflexible, untrainable, disorganized, undisciplined, and cannot function as a team.

The other team clearly understands and applies its 30-page outline of process rules, and it has a comprehensive repository of efficient and effective process tools. However, at the moment the team is weak on the details regarding the use of power–wizzy–gui–whatever 1.5. This team is very quick to learn, reliably operates as a cohesive team, is adept with abstractions, is well-grounded in fundamentals of software architectures and data management, is highly flexible, self-motivated, and contains clever problem solvers.

You decide.

References

1. Shneiderman, Ben. *Software Psychology: Human Factors in Computer and Information Systems.* Little, Brown. Boston. 1980.
2. Curtis, Bill. Managing the Real Leverage in Software Productivity and Quality. *American Programmer,* Vol. 3, July-August 1990.
3. Schulmeyer, Gordon. The Net Negative Producing Programmer. *American Programmer,* Vol. 5, June 1992.
4. *Chaos.* The Standish Group. Dennis, Mass., 1995. Also at http://www.standishgroup.com/chaos.html
5. Paulk, M. et. al., Capability Maturity Model for Software, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, Pa. 1993.
6. Radice, R., *ISO 9001 Interpreted for Software Organizations,* Paradoxicon Publishing, 1995.
7. Washington Post classified ad website. 10/01/99, www.washingtonpost.com/wp-adv/classifieds/careerpost/front.htm

Note

1. If you have a preference for mathematics, you can assign values to each of the listed skill areas as follows: Essential–8 points, Important–4 points, and Useful–2 points. These values sum to 100 points.

About the Author

Dr. Richard Bechtold is a principal consultant who supports industry and government in the analysis, design, development and deployment of improved software management, engineering, acquisition, and risk-reduction processes. He assists clients in process analysis, modeling, definition, training, and deployment. Bechtold also assists government acquisition agencies in the systematic evaluation of contractor software process capability. He has more than two decades of experience in the software industry.

He is an adjunct professor for George Mason University, where he teaches software project management and software process improvement to masters and doctorate students. He has written more than two dozen publications relating to software project management, software process improvement, risk management, acquisitions, logistics, and related topics. His latest book, *Essentials of Software Project Management* was published in August 1999.

Bechtold received his doctorate degree from George Mason University.

Dr. Richard Bechtold
Abridge Technology
42786 Oatyer Cr.
Ashburn, Va. 20148-5000
Voice: 703-729-6085
Fax: 703.729.3953
E-mail: rbechtold@rbechtold.com
Internet: www.rbechtold.com

Call for Articles

If your experience or research has produced information that could be useful to others, CrossTalk will get the word out. We welcome articles on all software-related topics, but are especially interested in several high-interest areas. Drawing from reader survey data, we will highlight your most requested article topics as themes for future issues. In future issues of CrossTalk, we will place a special, yet nonexclusive, focus on:

Network Security
October 2000
Submission deadline: June 1

Software Acquisition
November 2000
Submission deadline: July 1

Project Management
December 2000
Submission deadline: August 1

Modeling and Simulation
January 2001
Submission deadline: September 1

Configuration Management
February 2001
Submission deadline: October 2

We accept article submissions on all software-related topics at any time; issues will not focus exclusively on the featured theme.

Please follow the Guidelines for CrossTalk Authors, available on the Internet at www.stsc.hill.af.mil.

Ogden ALC/TISE
ATTN: Heather Winward
7278 Fourth Street
Hill AFB, Utah 84056-5205

You may e-mail articles to features@stsc1.hill.af.mil.
or call 801-775-5555 DSN 775-5555.

Goal-Problem Approach for Scoping an Improvement Program

by Mary Sakry and Neil Potter
The Process Group

In this paper, we will explain an approach to scoping an improvement program based on problems and goals of the organization. By adopting this approach, organizations are able to make significant progress on real issues, and make progress on the process improvement model or standard they are trying to achieve.

This article is reprinted courtesy of The Process Group Post Newsletter. It first appeared in that publication's September 1999 issue, Vol. 6, No. 2.

The most common approach for process improvement we have seen during the last 10 years is to document all processes. We do not know exactly why people do this, but they do.

This approach is amplified when an organization rushes to adopt a sweeping solution such as ISO9001 or the Software Engineering Institute's (SEI) Capability Maturity Model (CMM®). In the light of a goal stating, "Be SEI CMM Level 3 by December," the approach of documenting all processes is reinforced, and might even appear natural.

A process-centric approach can work, but it has a high risk of failure. To be successful, it must involve individuals who can internalize how the process documents will be used before they are completed. This is a rare skill.

The goal-problem approach starts with a business goal and works backward to determine what improvement actions are necessary to achieve that goal. Here is an example.

During a client visit to help plan a process improvement program, we learned that the group was about to establish six teams to work on the six Key Process Areas (KPAs) of the CMM Level 2. We suggested that the developers and managers temporarily forget about Level 2 and state all of their major problems. Then they were asked to state the goals they were trying to achieve over the next six to 18 months. After one hour of discussion, they created a list (Figure 1).

The next step was to have the group compare the list of problems and goals with the topics of the CMM. In Figure 1 we have listed the related KPA names and activities in parentheses after each item. If the client had been using ISO9001 or The

Figure 1. *Problems and goals list*

Problems	Goals
1. Get better requirements. Requirements tracking not in place; changes to requirements are not tracked; code does not match spec. at test time. [Level 2: RM - activities 1, 2, 3]	1. Orderly Plans for Development. [Level 2: SPP - activities 2, 5, 6, 7, 8, 13, 14]
2. Management direction unclear for product version 2.3. Goals change often. [Level 2: RM - activities 1, 3, verification 1]	2. Understand what our capacity is—develop one list of all the work we have to do. [Level 2: SPP - activity 7, ability 1]
3. Hard to revise project plan—items drop off, new things get added, plan is out of date. [Level 2: SPTO - activity 2, 8, 9]	3. Improve schedule tracking and communication of changes to impacted groups. [Level 2: SPTO - activities 3, 4]
4. Wrong files (e.g., DLLs) get put on CD—don't know what the right ones should be. [Level 2: SCM - activities 4, 7, 8, 9, 10]	4. Successfully deliver Serial Number Tracking product. [Level 2: RM - activities 1, 2, 3, SPP - activities 10, 6, 13]
5. Defect repairs break essential product features. [Level 2: SCM - activities 5, 7, 6, 9, 10, abilities 1, 2, 4, 5, verification 3, 4]	5. Improve performance of mainline software product. [Level 2: SPP - activity 11, SPTO - activity 7].
6. Customers are unhappy. There are approximately 300 outstanding defects that have not been addressed. [Level 2: SCM - verification 1, RM - activity 3; Level 3: IC - activity 1]	6. Identify needed skills for new designers and hire/promote and train accordingly. [Level 3: SPE activity 3, ability 2]
7. Difficult to find time to do critical activities (product development) versus crisis activities. [Level 2: SPP - activities 4, 10, 12]	7. Identify tools to support software developers. [Level 2: SPP - activity 14; Level 3: SPE - activity1]
8. Lack of resources and skills allocated to software design. [Level 2: SPP - activity 10]	8. Keep making a profit. Keep customers happy. [Level 2: RM - activities 1, 2, SPP - activities 10, 12, 13, SPTO - activities 4, 6, 8, 10, SQA - activity 5, Level 3: SPE - activities 2, 7, IC - activity 1, PR - goal 2]
9. Quality department—need team training (product and test skills). [Level 2: SQA - abilities 2, 3, 4]	9. Identify tools to support software testers. [Level 2: SPP - activity 14; Level 3: SPE - activity1]
10. Changes to specifications and documentation are not communicated effectively to documentation and test groups. [Level 2: RM - activities 1, 2, 3, SCM activities 7, 5, 6, 9, ability 1]	10. Empower Quality Department to have final say on product shipment. [Level 2: SQA - activities 6, 7]
11. Unreliable project schedule estimates. [Level 2: SPP - activities 9, 10, 12, 5, 13, 14, ability 4]	**Definitions of SEI Level 2 Acronyms** **RM**= Requirements Management **SPP**= Software Project Planning **SPTO**= Software Project Tracking and Oversight **SQA**= Software Quality Assurance
12. Unclear status of software changes. [Level 2: SCM activities 8, 9]	
13. Testing does not necessarily comprehend things that matter to the customer. [Level 3: SPE activities 5, 6, 7]	

Malcolm Baldrige Award, we would have mapped the problems and goals to those documents.

What was the Improvement Program's Scope?

The scope was to address the problems and the goals of the organization. As you can see, 21 out of the 23 items (91 percent) map to Level 2. When all the problems and goals have been addressed, 46 percent of the Level 2 activities will have been addressed.

The key difference between this approach and addressing the six KPAs in parallel is that the problems and goals tell you which pieces of each KPA to address first. Regardless of the model or standard used, the problem-goal approach tells how to scope and sequence your improvement program.

Items Not Matching Improvement Model or Standard

In Figure 1, not all of the problems in the list closely match the areas of CMM Level 2. For example, there is not much in the CMM to specifically address goal No. 5. In this situation, you have to determine which areas are most important for the organization to fix now. Serious problems should be worked on first.

What is Learned from this Approach?

There are five significant lessons to be learned from adopting the goal-problem approach:

1. All process improvement can be meaningful.
2. The problems and goals help the organization identify which pieces of a model or standard to work on first. A model or standard should no longer be seen as providing an all-or-nothing approach, because this often leads people to do everything at once, regardless of whether it is appropriate. A model or standard can be treated as a large toolbox of little actions, ideas, and solutions useful at different times.
3. Any process document that is developed to solve a problem will be meaningful and useful. The process improve ment team will be less tempted to gold-plate the process, since its scope will be defined by a problem.
4. The group's motivation to work on improvement issues will increase. The improvements will be directed toward improving the group's ability to produce software. Barriers to success will be solved systematically.
5. An organization will be focused on solutions rather than process documents. Some of these solutions will involve processes; some will involve tools or behavior changes.

Using the Approach at a Project Level

Below is an example from a project at a different client. We asked the project manager for a significant project goal. From this goal we derived areas that needed improvement by asking two specific questions. These resulting problems formed the scope of the improvement program for this project.

What is your goal?

- Reduce release cycle to six to nine months.

What problems are preventing you from achieving the goal?

- Changing requirements.
- Loss of resources—difficult to replace people with specialized skills who leave the project.
- Too many features to put into a six- to nine-month development cycle.
- Poor quality of incoming code from other groups.
- Inadequate availability of test equipment.

What other problems do you have related to this goal?

- Lack of visibility within any life cycle phase—it is difficult to know whether we are ahead or behind schedule.
- Do not always have the resources available to complete the planned work.
- Difficult to find defects early.

We stepped through each of the answers and made a note of the KPA activity that could significantly help address the problem area. We recommended some of the more advanced Level 3 KPA components since this group was almost Level 2.

In this example, five out of the eight problems, or 63 percent, mapped to SEI Level 2, and 100 percent mapped to SEI Level 3. The scope of the improvement program should be the problems and goal. By addressing these, the project manager will make significant progress toward completing Level 2 and starting Level 3.

What questions helped you scope your improvement effort?

To scope the improvement effort, we asked the following questions:

1. State one goal for which you will be accountable over the next six to 18 months.
2. What prevents you from achieving this goal?
3. What other problems do you have related to this goal?
4. If you use a process improvement model or standard, which items help each of the problems listed? (Choose individual items at the detailed level, not large blocks of items.)

Figure 2.[1] *Goal: Reduce release cycle to six to nine months*

Problems	KPA component that would help this problem
Changing requirements.	Level 2: RM - activity 3, SCM - activity 5. Level 3: SPE - activity 2.
Loss of resources - difficult to replace people that leave the project due to specialized skills.	Level 2: SPTO - activities 2, 8. Level 3: TP - activities 1, 2, SPE - ability 2.
Too many features to put into six to nine month development cycle.	Level 2: SPP - activities 4,12,13. Level 3: SPE - activity 2.
Poor quality of incoming code from other groups.	Level 3: IC - activities 2, 5, 6, PR - activity 2.
Access to equipment to test code.	Level 2: SPP - activities 13, 14.Level 3: SPE - activities 6, 7.
Lack of visibility within any life cycle phase—it is difficult to know how much we are ahead or behind schedule.	Level 3: ISM - activities 7, 4, 11, verification 2. (ISM stands for Integrated Software Management.)
Do not always have the resources available to complete the planned work.	Level 2: SPP - activities 4,12, 13. Level 3: ISM - activities 3, 5, 10, 11.
Difficult to find defects early.	Level 3: PR - activities 1, 2, ability 1.

Addressing all Items in Model or Standard Used

One primary concern with this approach is that an organization will not address all of the items in the model or standard used, since there might not be goals or problems related to all items.

When the first set of problems and goals have been worked, the next step is to repeat the cycle and determine the next set of problems and goals. This new set can be compared to the remaining items in the improvement model or standard. Over a one- to three-year period, each section of the model or standard will be matched up with a problem or goal.

For example, in the beginning of SEI Level 2, there may be little benefit to working on the process audit activity within Software Quality Assurance, since few processes are being followed. However, the need to audit a process becomes apparent once it has been defined, used, and proven effective.

One client highlighted this with its software release management process. Performing an audit on the related Software Configuration Management (SCM) activities would have been futile before the release management had been improved. When SCM and release management were in place, one employee bypassed the process and incorrectly released a software patch to a customer by e-mail. The software did not work and the customer was furious. The need for SCM auditing became apparent. After the audits, the developers and managers realized that they had a mechanism to verify execution of the defined release management activities.

There will be situations where some items of the model or standard are not used when solving a problem or achieving a goal. These items should be left until the end of the improvement cycle. At that time, one of three scenarios usually occurs:

1. Outstanding items will be put to good use.
2. Items will be declared *not applicable.*
3. Items will be performed academically to meet the letter of the law. The focus should, of course, be on the first scenario.

Conclusion

Scoping an improvement program can be difficult and frustrating. The task becomes daunting when a process model or standard is adopted wholesale. However, a simple, immediately available solution exists. The goals and problems of an organization can provide a timeless and effective scope for any improvement program. An improvement model or standard can then be used as a source of ideas, solutions and actions to achieve this scope.

The resulting improvement program is compelling, practical, and focused on the goals and problems of an organization. Using this approach, it is easy to implement process improvement in a phased manner, which provides people with timely solutions aimed at their specific needs.

Note

1. Definitions of SEI Level 3 Acronyms:
   TP = Training Program
   SPE = Software Product Engineering
   PR = Peer Reviews
   IC = Intergroup Coordination
   ISM = Integrated Software Management.

About the Authors

Mary Sakry is co-founder of The Process Group, a company that consults in software engineering process improvement. She has 23 years of experience in software development, project management and software process improvement. For 15 years, she was a project manager and software engineer within Texas Instruments (TI) in Austin, Texas. In 1989, she worked on the TI Corporate Software Engineering Process Group TI to lead worldwide software process assessments. The last two years of TI were spent consulting and educating software developers and managers on software project planning, risk management, estimation, SEI CMM®, inspection and subcontract management. Sakry was the first SEI-authorized lead assessor for CBA-IPI process assessments. She has an master's degree in business administration from St. Edwards University, and a bachelor's in computer science from the University of Minnesota.

Neil Potter is co-founder of The Process Group. He has 14 years of experience in software design, engineering and process management. For six years, Neil was a software design engineer for Texas Instruments in Dallas, developing Electronic Design Automation software. The last two years at TI he was a Software Engineering Process Group manager consulting in the United States, England and India in the areas of software project planning, risk management, estimation, SEI CMM® and inspection. Potter is an SEI-authorized lead assessor for CBA-IPI process assessments. He has a bachelor's degree in computer science from the University of Essex in England.

The Process Group
Voice: 972-418-9541
Fax: 972-618-6283
E-mail: help@processgroup.com
Internet: www.processgroup.com

6th Annual JAWS S^3 is scheduled for June 25-30 in San Antonio, Texas

Over the years, the Joint Aerospace Weapon Systems Support, Sensors, and Simulation Symposium and Exhibition (JAWS S^3) has addressed target acquisition, the dirty battlefield, the electromagnetic spectrum and its impact on smart and brilliant weapons, and a host of other relevant topics.

JAWS S^3 will focus on the connectivity of various levels of modeling and simulation and their connectivity in support of this mission. JAWS S^3 2000 will feature senior-level decision-makers, who are in a position to impact the directions on these important defense issues, sharing their insights.

— Jim O'Bryon, Deputy Director, Operational Test and Evaluation/Live Fire Testing, Office of the Secretary of Defense

— E-mail: varmaa@navair.navy.mil for more information.

Four Rs of Software Process Improvement

by Johanna Rothman
Rothman Consulting Group Inc.

Process improvement projects can be difficult to start, keep on track, and assess results. We can use the same requirements gathering and specification techniques that we use on product projects on our process improvement projects. This paper discusses how to define requirements for process improvement projects and how to use reviews and retrospectives to assess the results of process improvement efforts.

Software projects are hard to do correctly; many of us have started process improvement projects to make our projects work better. Process improvement projects are not particularly easy either. I recently saw the following ad on the Web:

> *There is a bright future ahead on the right road to process improvement. We know the right road and all the steps in it to achieve process improvement success. Our experts will guide you, making sure every step is aligned with your company's goals and everyone in the organization is headed in the same direction.*

If only it were that simple. We could call in the experts, hand off our process improvement projects, and live happily ever after. Unfortunately, every process improvement project is different, because each organization has its own culture and problems.

I apply the following standard project questions to discover the motivation behind my process improvement project:

- What outcomes do you expect?
- What are the desired process improvement results?
- What do you need to do to know about those results?
- What are the process improvement requirements?
- How will you know you have achieved those results?
- What techniques will you use to review, test, and measure your process improvement project?

Use Results to Generate Requirements

Process improvement projects are supposed to create new capabilities in an organization—to affect *how* you perform the other projects in your company. What drives your process improvement project? What capabilities are you trying to change? Those new capabilities are your results.

Management does not always clearly state its desired results. Your managers may think their directives are the desired results. Look beyond the directives to find the real business results. I use context-free questions [1] to get at the results. Context-free questions might be:

Who are the clients of the process improvement project?

What does a highly successful solution look like?

What is that solution worth to you?

Why are these results desirable?

Following is an excerpt of a conversation I had with a senior manager (SM):

SM: We want to reduce cycle time.

JR: Why?

SM: We want to get our products to market faster, so we can improve our return on our product development investment.

JR: What return are you getting now, and what return do you want?

We discussed the total costs of development and support from some previous releases and came up with a specific desired business result:

Reduce cycle time in order to double the return on investment (ROI) from major releases.

This is very optimistic, but at least we understood the context of the changes we would have to consider.

If you receive directives for process improvement, discuss reasons behind the directives. Work with management to discover the business requirements, true desired results, and the context of your process improvement. Create specific and measurable requirements that make sense to everyone involved—management, the product development staff, and the process improvement project team.

All too often, directives for process improvement produce sneers and cynicism from the technical staff. They are cynical because they may not believe the managers are interested in process improvement. Sometimes technical staff perceive process improvement as a fad-of-the-month. Sometimes the technical staff is sure this is some other thing management has decided would be good for workers to accomplish instead of the product development work. When you define specific and measurable business results, you can avoid or minimize much of that cynicism.

Table 1 gives examples of how to convert mandates/directives to business results.

Table 1: *From Directives to Results*

Directive	Cynical reaction	Possible business reasons for process improvement and specific results
Be at Level x by y	Pass an audit. I wonder if those guys are from the IRS.	Introduce and maintain new project management capabilities. Be able to proactively manage projects, anticipate risks. These new project capabilities would increase our ability to ship product on time, and decrease customer support costs by half.
Reduce cycle time	They think we are wasting time, probably surfing the Web. They just want to make us work harder.	Our customers want us to ship more features faster. If we can get each release out faster, we can sell more products and increase our market share by 30 % in two years.
Ship more features	They want us to put the kitchen sink into every release. They think with process, they can make us put more features into each release.	Our customers want more features, but it is difficult to work on multiple releases in parallel. If we could define appropriate life cycles and development techniques to release more features per year, we could increase revenue by 40 % annually.
Reduce costs	Uh-oh. We know what costs they want to reduce—us!	We spend too much time on rework during each project and after the project ships. If we knew different ways to work, we could avoid generating patches and fixes, and work on new products. We could reduce maintenance costs by at least 50%, and put the money toward new product development.

Define Your Process Improvement Requirements

Once you have specific, measurable business results—the context of your process improvement project—you can determine the rest of the process improvement project's requirements.

If your organization fails to define and manage requirements, your process improvement effort will sputter and probably fail. To succeed, you must identify your requirements, verify them against your results, manage them through the project, and test the new process against the requirements. It is not simple, but it is the right way to improve.

Identify Your Process Improvement Requirements

After asking context-free questions, define the users, attributes, and functions [2] to identify the process improvement project's requirements.

Users

Who are the favored and disfavored users?[1] Are there primary or secondary users? Users might be the sponsoring management, potential assessors, the technical product development staff, and the product development management. You might decide that customers are not direct users of the process improvement project. You may decide to prioritize the needs of the primary and the other users. Identify the disfavored users, especially when you rank user needs. If you do not know who your disfavored users are and address their needs, your process improvement effort will fail.

One organization identified its technical staff as its primary users. The potential customer assessors are secondary users. They are creating a process that serves the technical staff first, and serves their assessors second. This organization does not appear to have disfavored users. They do not have coding cowboys, royalty, or busybodies who interfere [3] with the way work is done now.

Attributes

Do you want to be able to better predict how long your projects will take? Is project performance an attribute, such as being able to support a quarterly release? Is cost to market an attribute of your projects? How testable, visible, auditable, maintainable, controllable do your projects need to be?

Another organization decided that time to market, in the form of a quarterly release, was a crucial attribute for their products. When it planned its process improvement project, the organization knew it could not complete the effort in one quarter. It wanted quarterly deliverables from its process improvement project, in the same way it wanted quarterly deliverables from its product projects. Part of the organization's process improvement was culture change across the organization and across every project, including its process improvement project from long release cycles to repeatedly delivering small chunks of functionality on a quarterly basis.

Functions

What activities does your process improvement project need to perform? Many of those activities will be the same for your product projects as your process improvement projects: meet, communicate, negotiate, design, implement, examine, review, test, etc. Will it be a permanent ongoing project, or will it have an end? Should you bring in the functional groups across the organization, or only across product development?

One organization working on a concurrent engineering life cycle decided that it wanted its process to make certain the right people participated at the right time on the projects. It chose to have a specific design phase in the organization's process improvement project so it could discuss and test how to make design happen concurrently in its product projects.

Another organization made a business decision that it would not have any dedicated process people—the technical managers are responsible for defining and improving its product development processes. One attribute of the organization's process improvement activities is that it chooses to take one area for improvement and work on it for a year. That attribute affects some of the organization's functionality, project retrospectives. Each year, it holds a project retrospective where it reassesses its activities and results and decides on the next area of improvement. (I am not advocating this approach, but it appears to work for this organization.)

After identifying process improvement requirements, you can verify them.

Verify Process Improvement Requirements

If you do not verify your requirements, you will not get the business results you want. Verifying the requirements helps define issues and find defects in them.

Your business requirements define the context of your problem. The user/attribute/function description defines the whole customer or stakeholder problem. Review the process improvement context and problem together. If they align, you are set. If they do not, go back and talk to management and the rest of the people you have worked with so far.

Once you have reviewed the requirements within the process improvement project team, hold a technical requirements review with your management sponsor(s) and your process improvement team. Review the requirements with the technical staff, too, as a way to make the project more real to them, and to dispel any cynicism. Use your process improvement project to show the rest of the organization how early and how often review can improve the entire product.

Manage the Process Improvement Requirements

Process improvement projects are subject to the same kinds of pressures as product projects are. They are frequently under time or resource pressure, particularly if the staff is not dedicated to process improvement. Someone, somewhere, will want to change the requirements. Manage the process improvement project's requirements as you would manage other project change requests. The questions you ask for product projects apply here also:

- *What implications does this request have on the users, attributes, and functions?*
- *What are the schedule implications?*
- *Will the request change our ability to meet the desired business results?*

Test the Process Improvement Requirements

Every time you define or change the process improvement requirements, test them. I prefer to test process improvement requirements in a variety of ways:

- Verify that requirements meet all personnel levels of process improvement

Top Management	Does top management enable the process you are creating? Can it change what it does to make this new process work?
Middle Management	Does this process address how to do the work? Can the work be managed this way?
Technical Leads, First Line Managers, and Supervisors	Can they follow this process to make it work? Can they create a project plan with the process embedded in it?

Table 2. *Process improvement levels and definition*

and function [4] as in Table 2. I apply use cases to do this as part of the user-attribute-function matrix development and testing.

- Verify that requirements meet desired business results with technical reviews or walkthroughs with technical staff.
- Inspect the requirements for defects.

Choose the appropriate verification technique for your work products.

Role of Reviews in Process Improvement

All successful process improvement initiatives incorporate reviews. Process improvement is about changing the culture of your organization to achieve certain business results. Reviews reflect the culture of your process improvement project. When you perform walkthroughs, reviews, and inspections, you show the rest of the organization that you are willing to examine your work products. Reviews in your process improvement project will change the culture from where you are now, to one that is closer to an egoless programming culture [5].

Use your requirements verification activities to show how walkthroughs, reviews, and inspections can help product projects, as well as process improvement projects. Gain consensus on the requirements with technical reviews. Inspection is an appropriate technique to discover defects in your requirements, or in any of the other work products. Walkthroughs are excellent for training purposes.

One powerful approach to performing technical reviews on requirements is to ask what is missing and why. When you ask these questions, you send specific messages to the organization:

- You want to know what is not working so you can avoid it in the future.
- You are willing to question your work.
- You want to know problems and differences between the desired state and current state of product development.
- You are open to changing the culture of the organization.

You show you are willing to review your process improvement activities and process improvement project, and that you are open to change (you are not trying to be the process police). As process improvement staff, if you show that not only are you open to change, but that change is expected and desired, the rest of your organization may have less resistance [6]. Changes in results are the reasons for starting a process improvement project.

Results: How Do You Know You Achieved Them?

You have defined and verified your desired results. How do you know when you have achieved them? Are you meeting your users' needs? Are you meeting your defined attributes? Does your process improvement look successful?

Retrospectives

Process improvement projects can benefit from mini retrospectives as you proceed, and from major retrospectives as you achieve major milestones. What can you learn from your progress to date? Are you keeping your business results in mind as you difine your requirements? Are you testing the process with the technical staff, to verify it meets your requirements *and* their requirements. Are you working in ways you want to continue, or is there a better way to do your work? What are you haveing trouble changing and why? Is there something you have missed?

Retrospectives model the behavior you are looking for as part of the process improvement results: to look at what you have done, and improve on the past as you move forward. If you are not afraid to look back at your work and learn from it, the rest of the organization will be more likely to do so also.

Measurement

Aside from in-project and post-project reviews, you can measure results. If you are not sure how or what to measure, consider the goal-question-metric paradigm [7] to define your measures. The goals here are your business results that you derived from the context-free questions. From your results, ask questions that help you figure out when you have met the goal. Measure the answers to those questions. (See Table 3.)

Changing your process via process improvement is supposed to produce the desired results. Use measurements to track your changes and results, and keep your process improvement project on track.

Summary

Process improvement is all about changing the results our organizations currently have. These results will change our culture, so we have numerous stakeholders for process improvement projects.

Use your requirements elicitation and definition skills to derive desired results and the requirements that drive them. Use reviews as a testing mechanism for requirements and results, then ensure that all your stakeholders' interests intersect with your process improvement results [7].

Where appropriate, use your process improvement project to model product-project behaviors. Test pieces of your process on your project (sometimes called "eating your own dog food"). When you test the pieces, focus on the desired results.

Model the behavior you want your software projects to exhibit, especially the behaviors of requirements identification and management, and reviews. That will help you meet your desired results.

You will find that your right road to process improvement with requirements, reviews, retrospectives, and results.

Table 3. *Example of results management*

Goal	Reduce cycle time to double ROI from major releases.
Questions	For the last few major releases, what are current cycle times and the current ROIs? What size were the releases, and what was the staffing level? How much time did the staff work on these projects? Did the staff have to share time between projects? Is there anything else that affects ROI?
Measures	Track the last few releases of cycle time and ROI. For each release, staff effort by week over the project. (Include other measures for any of the other ROI related questions.)

Acknowledgements

I want to thank Brian Lawrence, Elisabeth Hendrickson, Karl Wiegers, and Jerry Weinberg for their valuable review comments.

References

1. Gause, Donald C., and Gerald M. Weinberg. *Exploring Requirements, Quality Before Design,* Dorset House Publishing, New York, 1989.
2. Lawrence, Brian. Requirements Happens, *American Programmer,* April 1997, Vol. 10, No. 4.
3. Rothman, Johanna. Retrain Your Code Czar, *IEEE Software,* March/April 1999.
4. Weinberg, Gerald M. *Quality Software Management, Vol.4: Anticipating Change,* Dorset House Publishing, New York, 1997.
5. Weinberg, Gerald M. *The Psychology of Computer Programming,* Silver Anniversary Edition, Dorset House Publishing, New York, 1999.
6. Wiegers, Karl. *Creating a Software Engineering Culture,* Dorset House Publishing, New York, 1996.
7. Wiegers, Karl. *Software Requirements: A Pragmatic Approach,* Microsoft Press, Redmond, Wash., 1999.

Note

1. Disfavored users work in a way that is counter to the system. Hackers, coding cowboys, and test gatekeepers are examples of disfavored users.

About the Author

Johanna Rothman observes and consults on managing high technology product development. She is the founder and principal of Rothman Consulting Group Inc., and is a member of the clinical faculty of The Gordon Institute at Tufts University, a practical management degree program for engineers. Rothman publishes *Reflections,* a quarterly newsletter about managing product development, and is the author of *Hiring Technical People: A Guide to Hiring the Right People for the Job.*

Rothman Consulting Group Inc.
38 Bonad Rd.
Arlington, Mass. 02476
Phone: 781-641-4046
Fax: 781-641-2764
Email: jr@jrothman.com
Internet: www.jrothman.com

The Determining Factor

by Doug Dynes
SBSI Consulting

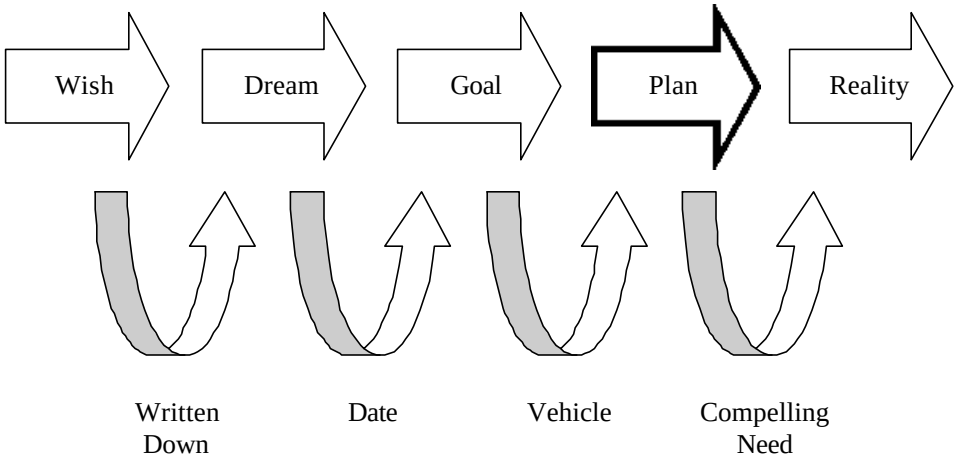
During the past six years in the process improvement consulting world, we have found one common factor in successful organizations. Not surprisingly, organizations that have failed share the same factor, or rather the lack of it. This determining factor is strong senior-level sponsorship. The Software Engineering Institute's Managing Technological Change (MTC) course demonstrates the power of sponsorship during the "roles" section. The course helps organizations identify key barriers to change efforts at any level. The senior-level sponsor is the "authorizing sponsor." This is the one person who can say "yes" to economic or strategic issues when everyone else says "no" and the effort proceeds anyway. This is also the person who can say "no" when everyone else says "yes" and the effort stops.

Throughout the many workshops, courses, and seminars we have given, sponsorship always becomes the main issue. If an organization has strong sponsorship, the chances for success increase exponentially. But a sponsor must do more than bless an effort, give it some funding, and direct the improvement group to make the project happen. A successful improvement effort needs a committed sponsor, not just an involved sponsor. The best way to define the difference between committed and involved is to make a comparison between a chicken and a pig in breakfast. The chicken is involved by giving its eggs and the pig is committed by giving its life. Senior-level sponsors need to exhibit committment.

Leadership is influence. A strong sponsor has to establish and foster a vision, define the compelling need, establish a management team that turns the vision into reality, and stick with it until the end. A sponsor needs to lead with vision, not manage by it.

Four initial building blocks are needed to start improving an organization: a mission, a vision, goals, and a strategic plan. Only the sponsor can move an organization from the strategic plan to reality. Far too frequently, organizations minimize the importance of a mission, a vision, goals, and strategic planning. Senior-level managers find these tasks to be unrewarding and to be in the way of doing their real work. But this is their real work.

The following model demonstrates a method of bringing an organization from the status quo, or an ad hoc environment, to an improved reality.



The first step in this process is the wish, or the original ideas and thoughts on how an organization can improve. A wish does not become a dream until it is written down. This is critical. Too often, sponsors think their organizations know what the sponsors want. Organizations will never know what their sponsors want until the sponsors tell them.

A dream becomes a goal when a date is given to it. Sponsors need to be leaders who are willing to draw the line in the sand. The organization's goals, coupled with a chosen vehicle, will yield the organization's strategic plan. This vehicle typically is

composed of a model used to measure progress, a process improvement group used to facilitate the change, and an established management steering group used to direct the organization's business. The authorizing sponsor needs to chair the management steering group.

The next step is moving the organization from the strategic plan to reality. Only a compelling need will move an organization forward, and only the sponsor can define the compelling need. This phase contains the real work. The sponsor will spend most of his or her time reinforcing the need for the change, which amounts to teaching the plan.

Many organizations fall far short of improved reality. Perhaps a strategic plan was completed, but it ended up as shelfware and now the organization has another failure attached to its culture. The key to avoiding this failure rests in the ability of the management steering group. This provides the leveling plane. If an authorizing sponsor is strong enough, he or she can force the change to happen at the expense of the organization's livelihood. We have seen firsthand where a sponsor forced the issue. The change occurred, but caused workers' divorces, early retirements, higher attrition, and severe medical problems. When the sponsor left, there was no support structure to maintain the pace, the improved reality evaporated, and the effort became another statistical failure.

An established and functioning management steering group is composed of the authorizing sponsor and his or her direct reports. This group meets on a regular basis. In this forum, all management decisions concerning any and all resources need to be made. In this instance, resources are defined as money, time, people, and the product's functionality. This meeting provides a forum where the direct reports inform the boss of the good, the bad, and the ugly. Counseling with the sponsor about a situation before it becomes a problem should always be welcomed. The easiest way to gain this insight is to review and show the status of all projects during the meeting.

The use of project status reviews in management steering group meetings has facilitated outstanding process improvement results with our customers. We have seen rapid changes for the better in general project management, quality assurance, and performance measures. These areas comprise most of the Key Process Areas from the Capability Maturity Model (CMM®). The most dramatic results are found in the actual management steering group members. They have found that when the executive leadership manages the organization proactively, they spend more time in the decision-making process and less time putting out fires.

The inspired change agent must obtain sponsorship for the organization's improvement effort. If the sponsor is unsure, the change agent should begin by educating the sponsor on the need for this. We have seen this education process take weeks. In other instances, it never happens. If the sponsor sees the need and is willing to act, you have won your sponsor. The change agent must be ready to provide a solid plan for implementing change. If the sponsor fails to see the need or is unwilling to act on the need, the effort will fail.

In conclusion, the main thrust of this article is about people, not technical issues. We live in a society where our culture is good at getting new technology into the market place, but is very poor at getting our culture to adopt that new technology as a way of doing business. Technology is not and never will be the problem. It is the people working with the technology that cause the problems. Solve your people problems and the technical problems will find a way of fixing themselves.

The senior sponsor is supposed to lead and direct people. If the sponsor would do that, there would be no compelling need for this article.

About the Author

Doug Dynes graduated from the U.S. Merchant Marine Academy with a degree in engineering systems design. He is a Lieutenant in the U.S. Naval Reserve and has served in the U.S. Naval Special Warfare program. Dynes has worked for the STSC sinch 1993 as a consultant and technology adoption coordinator. Recently, he became the senior vice president of SBSI Consulting. He is a certified Project Management Professional, SEI instructor and visiting scientist, and holds a U.S. Coast Guard Engineering license.

Voice: 801-775-5734
E-mail: Doug.Dynes@hill.af.mil

Lyles Assumes Command of AFMC

WRIGHT-PATTERSON AFB, Ohio—Gen. Lester L. Lyles assumed command in April of the Air Force Materiel Command at Wright-Patterson Air Force Base. He replaces retired Gen. George T. Babbitt. Babbitt had been AFMC commander since May 1997.

Lyles served as vice chief of staff of the Air Force in Washington, D.C. since May 1999. He began his career in 1968 as a distinguished Air Force ROTC graduate. He has served in a variety of assignments, including program element monitor of the Short-Range Attack Missile, Headquarters U.S. Air Force in 1974, and special assistant and aide-de-camp to the commander of Air Force Systems Command in 1978.

In 1981 he was assigned to Wright-Patterson Air Force Base as Avionics Division chief in the F-16 Systems Program Office. He has served as director of tactical aircraft systems at AFSC headquarters and as director of the Medium-Launch Vehicles Program and space-launch systems offices at the Space and Missile Systems Center. He became AFSC headquarters' assistant deputy chief of staff for requirements in 1989, and deputy chief of staff for requirements in 1990. In 1992 he became vice commander of Ogden Air Logistics Center, Hill AFB.

He served as commander of the Ogden ALC from 1993 until 1994, then was assigned to command the Space and Missile Systems Center at Los Angeles Air Force Base until 1996. He became director of the Ballistic Missile Defense Organization in the office of the Secretary of Defense in 1996.

Babbit was commissioned in 1965 through the Reserve Officer Training Corps program at the University of Washington. He trained as an aircraft maintenance officer and served as officer in charge of fighter flight lines in the United States, the Pacific and Europe. He twice commanded aircraft maintenance squadrons and was deputy commander for maintenance of a European F-15 wing. Prior to assuming command of AFMC, the general was director of the Defense Logistics Agency at Fort Belvoir, Va.

Other assignments included deputy chief of staff for logistics, Headquarters U.S. Air Force and director of logistics for both Headquarters Air Training Command and Headquarters U.S. Air Forces in Europe.

State of the Software Industry, part 2

Editor's Note: This is part two of an interview that taps the wit and wisdom of George Washington, John Adams, Benjamin Franklin, Thomas Jefferson, and Abraham Lincoln regarding the state of the software industry.

Q: Gentlemen, inspection technology has proven to provide high return on investment, yet it receives far less publicity and use than other glamorous technologies. Why is this?

Abe: *A peer review [jury] too often has at least one member who is more ready to hang the peers [panel] than the engineer [traitor].*

Ben: *Any engineer [fool] can criticize, condemn, and complain, and most engineers [fools] do.*

Abe: *You can fool all the people some of the time, and some of the people all the time, but you cannot fool all the people all of the time.*

Ben: *Wise men do not need advice. Fools will not take it.*

Q: Is process improvement here to stay or just another fad?

Thomas: *A little rebellion now and then is a good thing, and as necessary in the software [political] world as storms in the physical.*

John: *Process Improvement [America] is a great, unwieldy body. Its progress is [must be] slow. Like a coach and six, the swiftest horses must be slackened and the slowest quickened, that all may keep an even pace.*

Ben: *Each year, one bad habit rooted out, in time ought to make the worst project [man] good.*

Abe: *My great concern is not whether you have failed, but whether you are content with your failure.*

Q: Why are many organizations struggling with process improvement?

Abe: *At what point then is the approach of danger to be expected? I answer, if it ever reach us, it must spring up amongst us. It cannot come from abroad. If destruction be our lot, we must ourselves be its author and finisher.*

John: *The right of an organization [a nation] to kill a bad project [tyrant] in case of necessity can no more be doubted than to hang a robber, or kill a flea.*

Q: Are new technologies a panacea for the industry's ills?

Thomas: *Nothing can stop the man with the right technology [mental attitude] from achieving his goal; nothing on earth can help the man with the wrong technology [mental attitude].*

Abe: *The best thing about the future is that it comes only one day at a time.*

Ben: *He is the best engineer [physician] that knows the worthlessness of most technologies [medicines].*

Q: What do you know about the working conditions of software engineers?

John: *Engineers [I] inhabit a weak, frail, decayed tenement; battered by their [my]colleagues [the winds] and broken in on by janitors [the storms], and, from all I can learn, management [the landlord] does not intend to repair.*

George: *Few Software Engineers [men] have virtue to withstand the highest bidder.*

Q: The industry is quick to jump on new technology bandwagons. Managers in particular seem to treat testimony and hype as fact. Is this dangerous?

Abe: *I believe it is an established maxim in morals that he who makes an assertion without knowing whether it is true or false is guilty of falsehood, and the accidental truth of the assertion does not justify or excuse him.*

Thomas: *It is as useless to argue with those who have renounced the use of facts [reason] as to administer medication to the dead.*

Q: Given the chance, would you like to lead a software development project?

Thomas: *I have no ambition to manage [govern] software [men]. It is a painful and thankless job [office].*

—Gary Petersen, Shim Enterprise Inc.

Published by the Software Technology Support Center

S T S C

Sponsored by the Computer Resources Support Improvement Program (CRSIP)

CrossTalk
Ogden ALC/TISE
7278 Fourth Street
Hill AFB, UT 84056-5205

BULK RATE
US POSTAGE PAID
Permit No. 481
Cedarburg, WI

F-22 Avionics Integration On Track

by Robert Barnes
Boeing Vice President and F-22 Program Manager

One of the most challenging milestones for the F-22 program this year is to begin flight-testing the Block 3.0 avionics system on the F-22 before the end of the year. We must meet this milestone before the Defense Acquisition Board will approve low-rate initial production—and we will.

Boeing, tasked with integrating the F-22's highly sophisticated avionics, is working alongside our teammates to ensure Block 3.0 is adequately tested in our Avionics Integration Lab (AIL) and on our 757 Flying Test Bed (FTB) so testing aboard the F-22 can begin on schedule.

We are confident we will meet this critical milestone. In fact, every software delivery Boeing has made to date has been on or ahead of schedule. Both supporters and critics have been closely watching our avionics testing this past year, and we welcome their scrutiny. The F-22's avionics are being tested thoroughly, and have been through more rigorous testing than any previous fighter at a similar stage in its development.

It is important to note that successfully flying the integrated avionics on the FTB more than one year prior to the actual F-22 is unprecedented in military aircraft development programs.

By utilizing our AIL and FTB, we are helping reduce avionics risks and contain development costs by enabling extensive evaluation and troubleshooting before full avionics are ever installed on the F-22. To date, the avionics have undergone more than 15,000 hours of testing in the AIL and 427 hours on the FTB.

The F-22 avionics test concept is progressive in nature, beginning with component-level testing, continuing with subsystems integration and verification at teammate and supplier sites, and finishing with verification of the full avionics system installed in the F-22. Modeling tools and system-performance simulations have been used for integration to maximize the efficiency of other types of testing. These models are based on system design and are being updated during integration to more accurately reflect actual system performance.

Systems are integrated in the AIL, where performance is initially tested. AIL testing includes functional and performance testing of the integrated avionics suite, and includes integration testing of the avionics operating with other on-board systems. Testing in the AIL is accomplished in two types of configurations. One has F-22 hardware and software, with tower-mounted antennas for open-air stimulation. The other also has real hardware and mission software, but also simulated sensors and environment.

Boeing has invested in high-fidelity open-air dedicated targets (airborne and ground) to support this testing. Together these configurations test the system to provide a high confidence of success at the next integration level.

Avionics development transitions into the dynamic open-air environment, first on the Boeing FTB and then on F-22 aircraft. FTB testing takes place in Seattle-area locations, with deployment to Edwards Air Force Base, Calif., operating areas to take advantage of range assets such as simulated threat emitters and military target aircraft.

The F-22 avionics suite is being developed in an incremental block-build fashion in order to break the avionics development effort into manageable segments. This approach reduces program risk by starting with basic functionality and progressing to more complex functionality as the avionics system matures through each successive software block release.

Complex capabilities and functions can be developed and thoroughly tested in separate incremental builds without impacting or being impacted by other functions. This approach allows greater flexibility in the test program.

Boeing has been testing the software blocks in its labs since early 1998. Block 1.1, which provides initial integrated avionics capability, was delivered ahead of schedule to Lockheed Martin in May 1999. Block 1.1 included 80 percent of the final F-22 hardware configuration and more than 900,000 lines of code.

The performance of the sensors and the closed-loop tracking function are crucial to the success of the integrated avionics system. Blocks 2 and 3 will focus on these areas early in the test program. They will also continue the build-up of overall avionics functions.

Block 2, which adds basic missile and electronic warfare functions, provides the initial capability of multi-sensor fusion. Block 2 has been integrated and tested in the AIL and AIL and has begun testing on the FTB. This provides additional early testing on real hardware with real apertures.

Block 3 will add additional radar and electronic warfare modes (more sensor fusion capability) in support of closed-loop tracking. Block 3 is being integrated in the AIL and will be tested aboard the FTB in September before delivery to Lockheed Martin. Block 3.1 will be delivered in June of 2001 and add additional weapons capability.

The F-22 team's low-risk avionics development approach blended with state-of-the-art software development tools and processes has proven successful. Boeing also is leveraging the company's core competence in large-scale systems integration-many of the integration tools and techniques that are being used are the result of lessons learned on the B-2, AWACS, 777 and other large airframe programs.

Thanks to excellent designers and modern software engineering techniques, considerably fewer anomalies have been encountered compared to previous programs. Those anomalies are more easily fixed due to the expertise of our system integrators. Overall, the avionics software packages have been performing exceptionally well. We are confident that performance will continue when we transition to the next stage of testing aboard the F-22.

About the Author

Robert Barnes was named Boeing vice president and F-22 program manager in February 1997. Based in Seattle, Barnes is responsible for Boeing work on the F-22 air superiority fighter. Boeing is teamed with Lockheed Martin to design and build the F-22 for the U.S. Air Force. Since joining Boeing in 1977, Barnes has held a variety of positions in engineering and operations management. Prior to his current assignment, Barnes was F-22 airframe product manager and operations director. In that role, Barnes managed all F-22 structures design and production requirements, including manufacturing, quality assurance, procurement and facilities. Before joining the F-22 program in 1992, Barnes was deputy program manager for the 777 Composite Empennage program beginning in 1989. From 1986 to 1989 he served as manufacturing services manager for the Boeing Composites Fabrication Center in Seattle and division manufacturing engineering manager. Other Boeing assignments include operations manager for the Navy A-6 attack-plane rewing program, manager of engineering and operations computing for the B-2 bomber program, and engineering computing systems manager in the Boeing Commercial Airplane Group. Barnes has extensive management background in composites, tool engineering, N/C programming and production. Born May 27, 1941, he attended the Georgia Institute of Technology and Washington University. His educational background is in mechanical engineering and industrial management.