CrossTalk
December 2000 The Journal of Defense Software Engineering Vol. 13 No. 12

Overcoming Obstacles Through Project Management

Project Management

Software Engineering Technology

Open Forum

wi crossed es

Due to a printing error, some of Dave Putnam's article *Avoid Self-Inflicted Wounds* … in the October 2000 issue was lost. The complete version can be found on the Web at www.stsc.hill.af.mil

Departments

CrossTalk

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

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

Subscriptions: Send correspondence concerning subscriptions and changes of address to the following address. You may use the form on page 31.

Ogden ALC/TISE
5851 F Ave., Bldg 849, Rm B-04
Hill AFB, Utah 84056-5713

Article Submissions: We welcome articles of interest to the defense software community. Articles must be approved by the CROSSTALK editorial board prior to publication. Please follow the Author Guidelines, available at www.stsc.hill.af.mil/CrossTalk/xtlkguid.pdf. CROSSTALK does not pay for submissions. Articles published in CROSSTALK remain the property of the authors and may be submitted to other publications.

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

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

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

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

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

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

Other Disciplines Lend Ideas to Project Management

Most CROSSTALK readers have acquired years of school learning to respect the structure provided by the academic subjects that underpin software development. As Einstein said, "You remember the magnificent structure, on the lofty staircase of which you were chased about for uncounted hours by conscientious teachers." Like Einstein's departure from Euclidean geometry, the articles in this edition depart from conventional approach, and show how techniques developed in other disciplines can be adapted to software project management.

In *Project Clarity Through Stakeholder Analysis*, Larry Smith of the Software Technology Support Center describes the need for stakeholder analysis and illustrates techniques with case studies. Successful management of software projects requires a keen awareness of the environment, selection of the right tools and team for the situation, and leadership that aligns project objectives and activities with stakeholder expectations.

Walt Lipke and Mike Jennings of the Oklahoma City Air Logistics Center merge techniques from statistical process control and earned value data to analyze project performance and improve cost and schedule estimates. Arlene Minkiewicz, provides a primer on software estimating basics then shows how software cost estimating tools can help an organization mature project planning, process focus, and quality management key process areas. Louis Poulin, president of GrafP Technologies Inc., applies system safety engineering principles to anticipate problems and prevent them from occurring in information technology projects. Each of the above borrows concepts from more traditional disciplines to develop methods to improve software project management.

Donald Reifer, Jeff Craver, Mike Ellis, and Dan Strickland address the question of the year in their article, *Is Ada Dead or Alive Within the Weapons System World?* They summarize the results of a study to shed light on the long-term viability and staying power of Ada. Comparisons of Ada and C/C++ compiler/tool availability and training support are shown along with discussion of current trends.

Paul Kimmerly then describes the leadership required to keep the "wind in the sails" while on the journey from the project centered CMM® Level 2 view to the broader organization-wide view of a Level 3.

Finally, in BACKTALK, Dr. Dave Cook and Les Dupaix combine the art of poetry with science of software development for a humorous look at the world of defense acquisition programs.

Collectively this month's articles communicate the need to go beyond the basics to creatively use fundamental principles from a variety of disciplines optimized for the situation. As director of the Air Force's Computer Resources Support Improvement Program, I am privileged to work with software professionals exploring new approaches to development, maintenance, and management of software intensive systems. Their creativity and intellect has paid off—more mature organizations, efficient tools, and a better bottom line. Air Force process improvement has delivered a return on investment on the order of 15:1 while improving quality. I hope this CROSSTALK spawns ideas that help make your software project successful.

Glenn A. Palmer

Lt. Col. Glenn A. Palmer
Director, Computer Resources Support Improvement Program

Project Clarity Through Stakeholder Analysis

Larry W. Smith
Software Technology Support Center

A project is more likely to be successful if it begins well. A good beginning includes time at the outset to discuss project stakeholders' key needs and expectations. This should be augmented with a documented plan to meet these requirements, deal with potential risks, and define project information communication routes to stakeholders. This paper outlines a simple stakeholder analysis method, describes specific case studies, and discusses additional project activities that benefit directly from the analysis.

Understanding, extracting and solidifying documented project requirements is one of the most difficult tasks. Often the customer must first be taught how to give clear requirements. Project managers and personnel frequently compound the issue by automatically assuming requirements will change; yet, they fail to plan for, or proactively anticipate changes.

Requirements go beyond hard and fast product technical specifications. It is equally tough to satisfy each end user's definition of functionality in delivered products. Also, often forgotten project requirements dealing with "softer," human-oriented needs and expectations have the potential to make or break a project. For example, a technical sponsor may insist that certain information be relayed to them at definite times in a specific format during the project life cycle. Or, a project manager may need to fulfill requirements with key players outside of the project's environment. These are examples of a derived requirement that is primarily communications oriented. Not surprisingly, project managers spend a significant amount of time clarifying requirements for a variety of project participants and customers.

Each project has many interested internal and external parties or customers. Often these individuals change, or their interests change during different phases of the project. This may cause other technical requirements—assumed to be stable—to likewise change. Interestingly, there are a number of nontechnical requirements that usually never change, but are forgotten. For example:

- Each team member is required to know the project goals and their individual, specific role throughout all project phases.
- A financial sponsor assumes up front that his money will be effectively spent and that information of the project's progress at milestones will be communicated to them as requested.
- A functional manager must provide an expert for strategic planning activities who can be used in cost and schedule estimating activities.

These requirements ensure that project goals and individuals' roles are clear. They lend confidence to completing project objectives, fulfilling communication needs, and following priorities. Experiences have shown that when these requirements are not met, the project could possibly be terminated or suffer.

How can you reach an understanding on these types of requirements? The answer lies in discovering and then aligning project requirements with the communicated and non-communicated derived requirements of all parties interested in the project. Fulfilling project management requires focus.

"*Project management* is the application of knowledge, skills, tools, and techniques to project activities in order to *meet or exceed stakeholder needs and expectations* from a project. Meeting or exceeding stakeholder needs and expectations invariably involves balancing their competing demands among:

- Scope, time, cost, and quality.
- Stakeholders with *differing* needs and expectations.
- Identified requirements (needs) and *unidentified* requirements (expectations)." [1]

To manage and balance competing needs and expectations, we must first know what they are or from whom they come.

What Is a Stakeholder?

Used as a general term, *stakeholder* describes individuals, groups, or organizations that have an interest in the project and can mobilize resources to affect its outcome in some way. A formal definition of a stakeholder is: "Individuals and organizations who are actively involved in the project, or whose interests may be positively or negatively affected as a result of project execution or successful project completion." [4] Project stakeholders usually include the project manager, the customer, users, team members within the performing organization, and the project sponsor. However, there are more than just these few.

If we expand our perspective to include those that can make a claim—any claim—on attention or resources, now and in the future, that list can become quite large. Some can become *winners* or *losers* as a result of the project, or participate as intermediaries in project execution or product development. These stakeholders can hold individual views and objectives, which may differ and conflict with those of other stakeholders.

Not meeting the needs or expectations of just one influential and powerful stakeholder at a critical time can possibly ruin a project. *Who is that stakeholder, and when is that critical time?* Typically, very little time is taken to:

- Clarify who are the project stakeholders.
- Discover and align stakeholders' expectations and individual impact on the project.
- Outline requirements change processes knowing that requirements (i.e., needs and expectations) will likely change.
- Relate needs and expectations to risk planning and risk response activities.
- Conscientiously plan project communication strategies.

If this information is available and documented, it can be monitored and revisited as necessary throughout the project to diminish the tendency to focus solely on moving forward; thus, we forget that project expectations change, and that communication habits may need to be altered. Stakeholder analysis is a method that can help us tackle these issues.

Definition of Project Management	Project management is the "application of knowledge, skills, tools, and techniques to project activities in order to *meet or exceed stakeholder needs and expectations*" and balancing their competing demands.
Organizational Planning Tool	Stakeholder analysis is a success oriented technique: "The *needs of the various stakeholders should be analyzed to ensure* that their needs will be met."
Project Plan Development	"Every *stakeholder has skills and knowledge* which may be useful in developing the project plan. The project management team must create an environment in which the *stakeholders can contribute appropriately*."
Project Organization	"The nature and number of project *stakeholders will often change* as the project moves from phase to phase of its lifecycle...techniques effective in one phase may not be effective in another."
Project Plan Updates	When making modifications to the project plan (including all sub-plans), "appropriate *stakeholders must be notified as needed*."
Scope Statement and Verification	Successful project managers ensure that stakeholders have *common understanding and acceptance of project scope.*
Project Cost Management	Successful project managers consider the information needs of stakeholders since "different *stakeholders may measure project costs in different ways* and at different times."
Quality Planning	The project management team "is responsible for ensuring that the *project stakeholders are fully aware*" of the organization's quality policy.
Project Team Directory	Communication is enhanced when there is a published directory that is maintained and "lists all the project team members and other *key stakeholders*."
Team Building	Creating teams that succeed is a process of improving "*interpersonal relationships among key stakeholders*."
Communication Planning Tool	Project managers should carefully design the approach they use to communicate with their stakeholders: "The *information needs of the various stakeholders should be analyzed* to develop a methodical and logical view of their information needs and sources to meet those needs."
Information Distribution	A project manager must make "needed *information available to project stakeholders in a timely manner*...including responding to unexpected requests."
Risk Identification	In understanding project risks, a project manager should conduct "risk-oriented *interviews with various stakeholders* [to] help identify risks not identified during normal planning activities."
Risk Quantification	The threshold level of a potential project risk cannot be understood until there is an understanding of *stakeholder risk tolerances*. "Different organizations and different individuals have different tolerances for risk." An opportunity for one may be a threat to another.
Procurement Planning	In a procurement situation, the customer relation switches and the "buyer becomes the customer and is thus a *key stakeholder for the seller*."

Table 1. *Prevalence of the need for stakeholder analysis in the PMBOK Guide®, a standard of the American National Standard Institute (adopted September 21, 1999) and an IEEE Adopted Standard (IEEE 1490-1998). Note that all references are from [1] and italics added in some cases.*

Importance of Stakeholder Analysis

Stakeholder analysis typically refers to the range of techniques or tools used to identify and understand the needs and expectations of major interests inside and outside the project environment. Understanding the attributes, interrelationships, and interfaces among and between project advocates and opponents assists in strategically planning the project. Herein lies a large portion of project risk and viability, and ultimately the support that must be effectively obtained and retained.

On significant projects this endeavor requires a certain level of political astuteness or *street smarts.* You need an understanding of the internal project environment along with the entities and including interfaces extending into the external environment. This requires multiple skills to discriminate among project groups, help develop potential support coalitions, or if necessary, reduce the impact of unseen opposition. For example, during times of limited supply computer component vendors must be savvy when multiple buyers are competing against each other for a high market position.

Projects typically require human solutions to reach completion. Using the metaphor of a stage production, consider the benefit of visualizing not only the actors on the stage, but also the producers, financiers, stagehands, marketers, benefactors, and the ultimate customer—the audience that we wish to return night after night. The ultimate for our project would be to design a similar script and accompanying choreography to outline policy, identify existing and potential interactions among players, design interventions and negotiations, accurately predict risks and thresholds, and anticipate sources of conflict and cooperation.

Organizational and Project Spotlight on Stakeholders

Stakeholder analysis is often considered the first step in strategic planning activities on an organizational level. Here we allow (or force) our minds to layout a future business concept considering all parties' needs in addition to our own. If stakeholder analysis is a valued and consistent activity at the organizational level, then its thrust can be felt on the project level. Attitudes and results can also filter down and be applied to multiple projects.

Stakeholder awareness and the need for analysis is prevalent among project management principles and accompanying artifacts. For example, its application is found throughout every project management knowledge area of a leading, industry-accepted project management standard, *A Guide to the Project Management Body of Knowledge* (referred to as the *PMBOK Guide*®), published by the Project Management Institute (PMI®). Table 1 outlines some of the definitions of terms or processes within the *PMBOK Guide*® showing the need for proper stakeholder awareness.

It becomes obvious that an understanding of stakeholders' needs and expectations is crucial to success. "The project management team must identify the stakeholders, determine what their needs and expectations are, and then manage and influence those expectations to ensure a successful project."[3]

Stakeholder Analysis Approach

When should stakeholder analysis be accomplished and by whom? Although worthwhile throughout the project to reassess key issues, particularly when the project is in trouble, stakeholder analysis is best accomplished before a project is initiated or at some beginning phase. The team should be aware of sensitive

information and maintaining confidences. Team members should be trustworthy and careful in dealing with information.

The following sections outline a simple stakeholder analysis approach. The first few stages may be sufficient for small projects with few stakeholders. Time spent doing the analysis should match project type and complexity. A few hours may be sufficient to clarify project objectives, key assumptions, and risks.

1. Identify project stakeholders. To be classified as a stakeholder, the person or group must have some interest or level of influence that can impact the project. Stakeholder interests must be understood, along with understanding potential project impact if a need is not met.

The first effort should be a brainstorming activity with appropriately selected members and an optional facilitator. All stakeholders should be initially considered and possibly dropped in later stages of the analysis. It is often difficult to force classifications into groups and determine who is considered truly inside and outside the project context. To gain a more powerful understanding of needs and expectations, it is usually helpful to identify these stakeholders by name rather than generic terms such as customer, owner, sponsor, etc. Figure 1 depicts an example of this high-level analysis using a notation similar to [3].

2. Identify stakeholders' interests, impact level, and relative priority. To refine the previous stage, stakeholders should be listed in a table or spreadsheet along with their key interests, potential level of project impact, and priority in relation to other stakeholders. Be careful to outline multiple interests, particularly those that are overt and hidden in relation to project objectives.

The key is to keep in mind that identifying interests is done with stakeholders' perspective in mind, not your own. This is difficult as interests are usually hidden and may contradict openly stated aims. Each interest should be related to the appropriate project phase; that is, interests change as the project moves from beginning to ending phases. With some stakeholders it may be crucial to extract interests by formally asking them questions such as:

- What are your project expectations?
- How do you benefit from successful project completion?

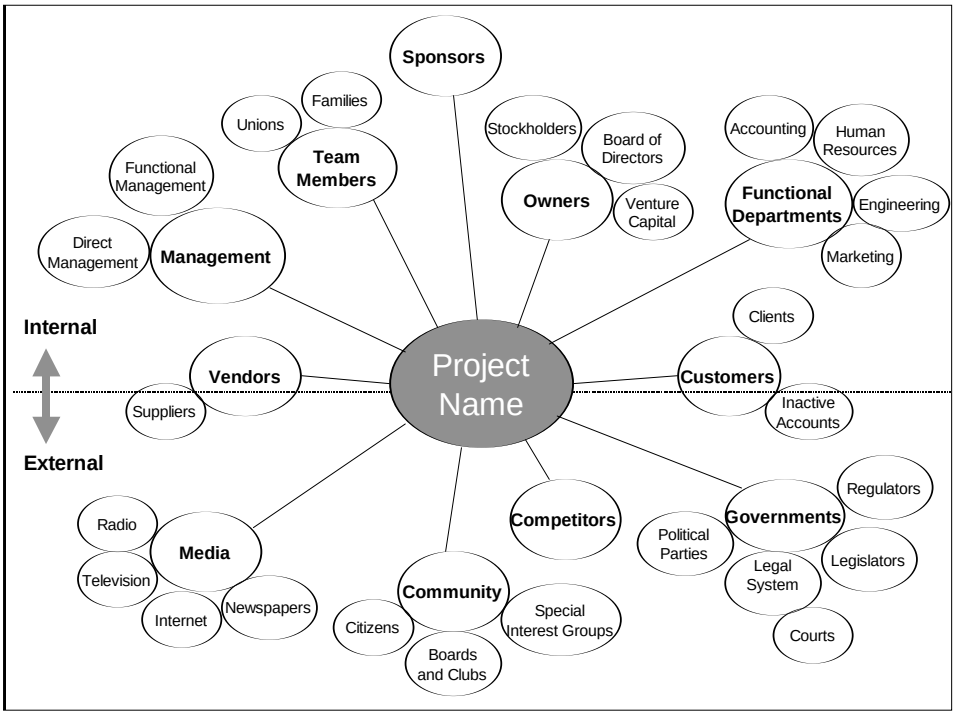


Figure 1. *Example of a stakeholder analysis context diagram.*

- Which stakeholders do you believe are in conflict with the project interests?
- Do stakeholders have contradictory interests?

Once major interests are identified, it is also useful to outline how the project will be impacted if these are or are not met. In most cases, a simple annotation of positive (+), negative (-), or unknown (?) can be used as well as high (H), medium (M), low (L), or uncertain (?). To align project success criteria with interests, an additional step is to give a rough prioritization of each stakeholder with their accompanying interests. Since not all needs can be met with the same level of intensity or at the same time, a prioritization schema would be beneficial. Table 2 provides an example of this information [4]. When this information is discussed in facilitated brainstorming sessions, flipchart paper and sticky-notes are typically used until formally documented.

3. Assess stakeholders for importance and influence. Determining whether stakeholders in a position of strong influence hold negative interests may be critical to project success. This level of understanding can best be reached by conduct-

Table 2. *Stakeholder interest and impact table.*

	Stakeholder	Interests	Estimated Project Impact	Estimated Priority
Primary	Owner	Achieve targets Liability (avoid at all costs) Increase sales margin	Med + High - Med +	1
	Sponsor	Successfully addresses needs of adjunct customer Appears competent among peers Provides new market to expand ventures	Low + Low - Med +	3
	Team Members	New product excitement Demand end-of-year bonus Retain and expand skill level Strike (if basic demands aren't met with new process)	Med + ? Med + High -	2
	Project Manager			
Secondary				

ing a formal assessment of each stakeholder's level of importance and influence to the project.

Influence indicates a stakeholder's relative power over and within a project. A stakeholder with high influence would control key decisions within the project and have strong ability to facilitate implementation of project tasks and cause others to take action. Usually such influence is derived from the individual's hierarchical, economic, social, or political position, though often someone with personal connections to other persons of influence also qualifies. Other indicators identified in [3] include: expert knowledge, negotiation and consensus building skills, charisma, holder of strategic resources, etc.

Importance indicates the degree to which the project cannot be considered successful if needs, expectations, and issues are not addressed. This measure is often derived based on the relation of the stakeholder need to the project's goals and purposes. For instance, the human resources department may be key to getting the project new resources at a critical time, and the accounting department key to keeping the finances in order and the project manager out of jail. The users of the project's product or service typically are considered of high importance.

These two measures, influence and importance, are distinct from each other. A project may have an important financial sponsor that can shut down the project at any time for any reason, but does not participate at all in the day-to-day operations of the project. The combination of these measures provides insight not only into how stakeholders interact, but can help identify additional assumptions and risks.

A diagram of these relationships can be useful to understand potential risks and highlight groups of stakeholders whose needs can be addressed in a common manner. Figure 2 shows such a diagram. The interest–influence measures can be annotation with a range of numbers (0-10) or high (H), medium (M), and low (L). Note that stakeholders in the high influence-high importance quadrant would be considered key stakeholders. Although counter to typical approaches, this area is where we may need to focus our attention at times when the project is suffering rather than targeting individuals in the opposite corner quadrant.

Those that are in the low importance-high influence quadrant have the potential becoming a high project risk. For instance, an individual that does not have any apparent needs or provide any technical requirements to our project, but has undue influence over a key funding source, should be monitored carefully.

A more interesting picture would be a dynamic view over the life of the project rather than this static view. For instance, a key indicator of project success may be where the key customer is located at the conceptual, implementation, and closeout phases of the project.

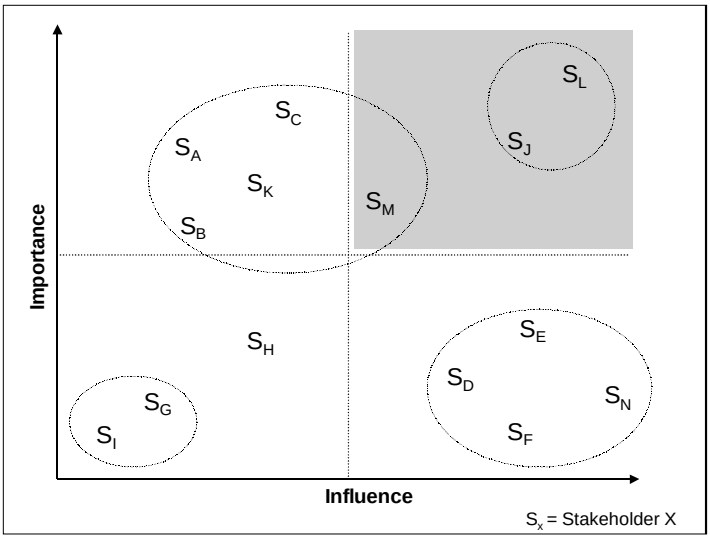
4. Outline assumptions and risks. Project success also depends on the validity of key assumptions and risks. In relation to stakeholders, risks are manifest when there are conflicting needs and expectations. For example, the interests of a stakeholder with high influence may not be in line with the project's objectives and can block a project's positive progression. To bring to light key risks, the project manager needs to clarify unspecified stakeholder roles and responsibilities, play "what-if" scenarios using unfulfilled needs and expectations, and double check the plausibility of assumptions made. Table 3 provides an example of documenting assumptions and risks in relation to key stakeholders. Note that a spreadsheet could be used to capture this information as well as that indicated in Table 2 and Figure 2.

This information provides a critical portion of a project's *risk management plan.* Using classical risk management methods, the project team could add another column in Table 3 for their pertinent risk mitigation strategies and action plans [6, 7].

Table 3. *Importance-Influence classification with assumptions, risks.*

Stakeholder	Estimated Project Influence	Estimated Project Importance	Assumptions and Risks
Owner	Low (2)	High (9)	Providing all the resources, but don't appear to have specific requirements.
Sponsor	High (10)	Medium (6)	We don't really know if the funding in the out years will continue. Has the propensity to change mind at any moment.
Team Members	Low (3)	Medium (5)	Appear to be happy with new process and systems equipment. Strike threats supposedly have decreased. Received numerous requests for additional training.
Project Manager			

Figure 2. *Importance–Influence Classification*



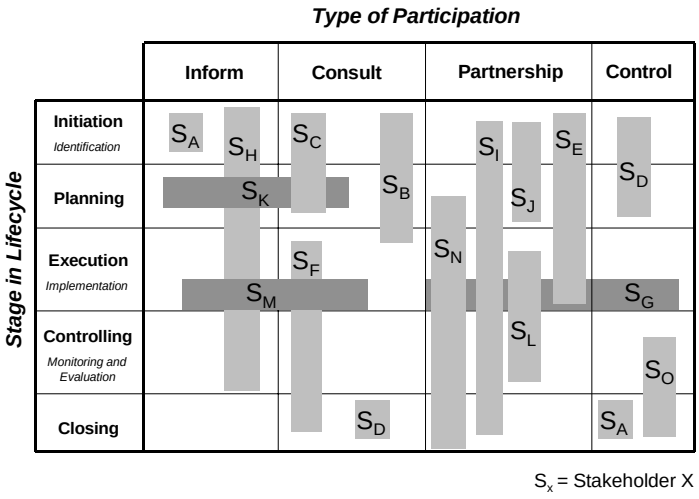
5. Define stakeholder participation. Now that we have made an effort to understand the stakeholders, we need to assess their level of participation and information needs. A well-designed project will not only clarify key stakeholder roles, but will define as much as possible who participates when. Not all stakeholders need to be involved in all aspects of the project in all life cycle phases. Previous analysis has helped us identify potential groupings of stakeholders. Similar individuals may have similar project information needs. We can use this information to reduce project report development costs and accompanying communication costs.

The participation matrix shown in Figure 3 is a method outlined in [5] that can assist project managers in categorizing their strategy for involving stakeholders. The life cycle stages should reflect the phases of the project (those shown are from [1]). Likewise, the types of participation shown are generic and should reflect those desired by the project team.

For example, Stakeholder A has been identified as someone who we want to inform at the beginning of the project and then, if possible, not involve them until the end of the project but in a more active manner. Stakeholder E is a subject matter expert that we want to maintain a close partnership with during critical project phases because of this advanced design experience. Stakeholder G is skilled as a facilitator conducting peer review sessions that we wish to hire during the product design phase.

Although a relatively difficult set of data to analyze and document, this information can be used to further highlight assumptions and risks. For instance, a project will be endangered with multiple key stakeholders all wishing to participate in project controlling functions. This matrix can be overlaid with the stakeholder information requirements (type, frequency, and format) to assist in developing the *project communication plan.*

Figure 3. *Stakeholder participation matrix.*



Sample Case Studies

This technique has been used on a number of projects differing in application area, duration, and complexity. Two projects are described here. They have been simplified to allow presentation of key concepts.

Case Study A: Where Is the Customer?

Case Study A describes a two-year project involving large teams in the banking industry that fortunately has not yet been completed. At the outset of the analysis (that started in the beginning of the second year), it was clear what the key risks were. Team members had become frustrated with the project for primarily two reasons: (1) functional managers continually added secondary projects to their plate, and (2) project requirements never seemed to be clear. The analysis showed that the primary customer was never brought into project planning discussions, the project team members were not encouraged to talk with customers, and the precisely defined secondary set of project requirements did not really belong to anyone.

Figure 4 highlights this situation in the project implementation phase. Key points to note are: The primary customer was deemed of little importance; the secondary customer went from being important in the design phase to not existing in the implementation phase; The functional managers wielded too much power in the matrix environment of the project. To improve the

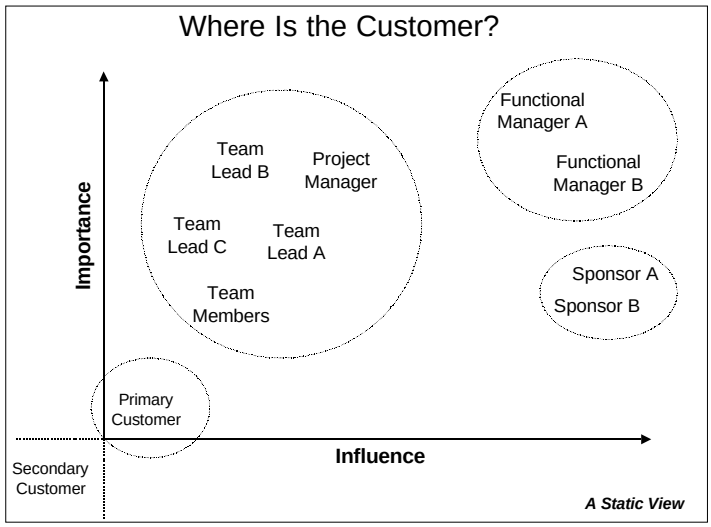


Figure 4. *Case Study A.*

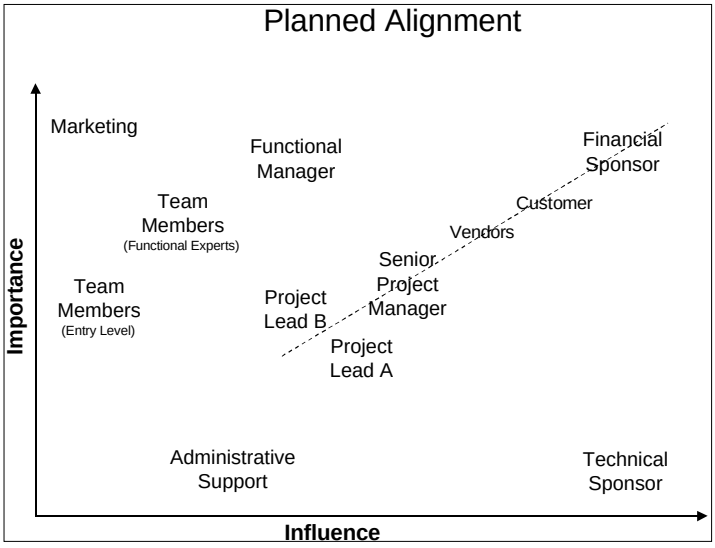
chances of success, the project manager realized that he must have both project sponsors more on the side of the project, and tactfully convince them to help the functional managers understand that they were ruining the project.

Case Study B: Planned Alignment

Case Study B describes a four-year project involving an international technical sponsor and a remote financial sponsor. The project manager understood well that this project would not work well unless all parties understood their roles and responsibilities. A relatively large amount of time was spent planning and defining the activities of the project and who would accomplish them. Furthermore, the customers and vendors had to be involved in all phases of the project, particularly at the beginning. Figure 5 shows a static view of the project taken at the middle of project execution. From the data collected there has been some shifting of key stakeholders, but the alignment has roughly stayed the same, at least according to the project team. Marketing concerns were not really known until a recent product review meeting where the representatives loudly expressed their apprehension at a potentially delayed shipping date.

The project team considered its communications strategy key to the success of its project. Although there have been some prob-

Figure 5. *Case Study B*



lems along the way (particularly growth issues), design and implementation barriers were made known before they became critical.

Conclusion

Stakeholder analysis is a technique that can help project team members understand the variety of stakeholders that have an interest in the project, and the individual nuances that can affect project risk. In an environment where organizational or office politics often appear to cloud a project's progression, stakeholder analysis provides the team with views and some basic measures that can help uncover and remove barriers. Consequences of completing such an analysis include information for additional tools that could prove valuable for the project team: a communication management plan, a risk management plan, and possibly a set of previously unspoken project requirements.

The technique described here compels project leaders to identify and support the interests of the key individuals or groups. When interests that cannot be supported arise, the knowledge that they exist and what level of influence the stakeholder may impose can be a great asset to the project team. The difference between success and failure can be simply knowing project advocates and opponents, understanding their respective needs and levels of influence, and aligning the project accordingly.◆

References

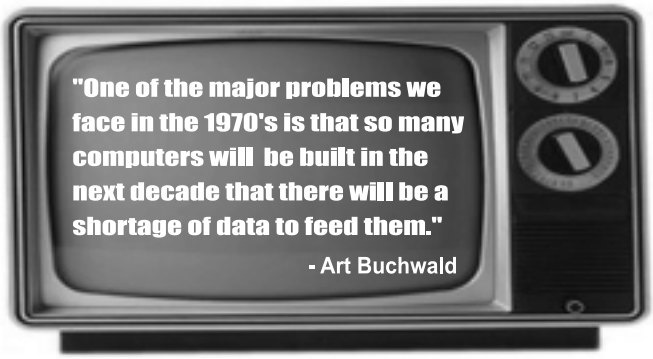
1. Project Management Institute (PMI®), *A Guide to the Project Management Body of Knowledge*. Four Campus Boulevard. Newtown Square, Pa. 19073-3299. 1996. "PMI" is a trade and service mark of the Project Management Institute, Inc., which is registered in the U.S. and other nations. "PMBOK" is a trademark of the Project Management Institute, Inc., which is registered in the U.S. and other nations.
2. IBID.
3. IBID.
4. Cleland, David I. (ed), *Field Guide to Project Management*, John Wiley and Sons, 1998.
5. Overseas Development Administration, *Guidance Note on How to Do Stakeholder Analysis of Aid Projects and Programmes*, Social Development Department, July 1995.
6. *Continuous Risk Management Guidebook*, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, Pa., 1996.
7. Williams, Ray C., Pandelios, George J., and Behrens, Sandra G. *Software Risk Evaluation (SRE) Method Description (version 2.0)*, Software Engineering Institute, Carnegie Mellon University, CMU/SEI-99-TR-029, Pittsburgh, Pa., 1996.

About the Author

Larry W. Smith is a software engineer for the Software Technology Support Center at Hill Air Force Base, Utah. He has provided software engineering, software process improvement, and project management consulting for the Air Force and DoD as well as commercial and nonprofit organizations. He has also authored and managed technology evaluation contracts in support of these efforts. Smith has a bachelor's degree in electrical engineering from the University of Utah, a master's degree in computer science from Utah State University, and is a certified project management professional (PMP®). He is a faculty member at the University of Phoenix and provides PMP® certification instruction.

Software Technology Support Center
OO-ALC/TISEA, 7278 4th Street
Hill Air Force Base, UT 84056
Phone: 801-777-9712
DSN: 777-9712
Fax: 801-777-8069
E-mail: larry.w.smith@hill.af.mil

Q"U"O"T"E M"A"R"K"S



"You often don't really understand the problems until after the first time you implement a solution. The second time, maybe you know enough to do it right. So if you want to get it right, be ready to start over at least once."
- Eric S. Raymond, *The Cathedral and the Bazaar*

The best way to predict THE FUTURE is to invent it
- Alan C. Kay

"Adding manpower to a late software project will, more times than not, make it even later."
- Jean Howe

Software Project Planning, Statistics, and Earned Value

Walt Lipke and Mike Jennings
Software Division, Directorate of Aircraft Management, Oklahoma City Air Logistics Center

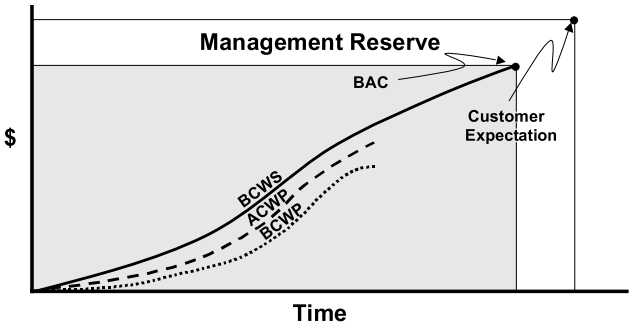
Increasingly, statistical methods are being applied to earned value (EV) data [1, 2]. In a previous publication [3] the author has discussed the use of statistical process control (SPC) with the earned value indicators and cost and schedule performance indexes. This application provides strategies and methods for gauging the performance of software projects and achieving project commitments. As an extension of managing performance, this article branches to using the statistical representation of the EV information to prepare project plans. Interestingly, the statistical approach yields not only the expected cost and completion date, but the management reserve required for an acceptable level of risk.

Nearly 15 years ago the Test Software and Industrial Automation branches of the Directorate of Aircraft Management began using EV methods to manage software development. During this time several refinements in the application of EV were made. The work breakdown structures have evolved and are much more sophisticated today than they were for the first few projects where EV was applied. The planning and scheduling practices have improved tremendously from simple paper and pencil tabulations to the use of commercially available automated tools. With these tools *what if* scenarios can now be performed during planning to account for possible risk areas. Along with these improvements, the ability to predict project outcomes and to strategize needed recovery actions was significantly enhanced three years ago by applying EV cost and schedule performance indexes [4]. Project managers now have a tool to help them choose the appropriate recovery strategy along with necessary actions.

Although these are significant improvements that have aided greatly in managing software projects to achieve the required performance at the negotiated cost and completion date, software organizations today are feeling pressure to apply the control chart method of SPC. Control charts began in the 1930s and were applied to manufacturing processes to maintain quality control of assembly line products. Control chart concepts and methods fell out of favor in the United States by the 1950s, but were revived in the 1980s. Dr. Edward Deming became an international celebrity from the impact control charts had on the success of the Japanese production of automobiles.

The application of SPC to software management is not very straight-forward; the automobile assembly-line application does not translate directly to software. The rate of software development is low, and none of the products are prepared to identical specifications. Yet a belief persists that SPC control charts must be used in order for software management to know that the development process is in control. Just as in manufacturing, if anomalous behavior occurs in the software process it should be recognized and corrected in order to minimize the impact on the delivered product.

Along with the rest of the software industry, we have struggled to develop a meaningful application of SPC control charts. EV indicators and cost and schedule performance indexes is proving very useful. In the publication [3] cited previously, the indicators were shown to be a management tool for statistically representing project performance. That paper also provided project managers with methods to overcome poor performance, and it alluded to further application in the areas of software project planning and process improvement. This paper addresses project planning, including the quantification of risk in both cost and schedule. Mitigating the risk with management reserve is included in the discussion.



BCWS = Budgeted Cost of Work Scheduled
ACWP = Actual Cost of Work Performed
BCWP = Budgeted Cost of Work Performed
BAC = Budget at Completion
CPI = Cost Performance Index [CPI = BCWP / ACWP]
SPI = Schedule Performance Index [SPI = BCWP / BCWS]

Figure 1. *Earned Value*

Earned Value

For this subject the book [5], *Cost/Schedule Control Systems Criteria, The Management Guide to C/SCSC*, by Quentin Fleming, is highly recommended for a more complete discussion of EV and its application to project management. For our specific application, an understanding of the EV indicators, cost performance index (CPI), and schedule performance index (SPI) is needed. To begin this discussion, refer to the point on Figure 1 labeled budget at completion (BAC). BAC is the performance expectation of the project; it identifies the cost and completion date for the project manager. Similarly, the point labeled customer expectation is the price and product delivery completion date promised to the customer. (The customer expectation is different from the planned project performance to allow for anticipated risk.)

EV management tools are based upon establishing a project baseline to achieve BAC. The project's performance is tracked against that baseline. The baseline performance is illustrated by the S-curve in Figure 1 marked BCWS, i.e., budgeted cost of work scheduled. The in-process performance tracking is facilitated by the two remaining curves shown: actual cost of work performed (ACWP) and budgeted cost of work performed (BCWP). BCWP is the earned value to date; it is a representation of the completion of project tasks and is traceable to the values allocated to project tasks during the planning phase.

During project execution, CPI and SPI provide information of performance efficiency. The CPI describes the rate of achieving

earned value with respect to the funding outlay. SPI is the rate of achieving earned value with respect to the schedule baseline, BCWS. These two indicators, taken together, have been shown to be a very useful software management tool [3, 4].

Statistical Process Control

There are several methods of performing SPC: scatter diagrams, run charts, cause and effect diagrams, histograms, bar charts, Pareto charts, and control charts [6, 7]. Although the other methods are useful, the application of control charts will be the only SPC application discussed in this paper.

As mentioned earlier, the question the software industry desires to answer is, "How do I know if my software development process is in control?" The answer is in the use of control charts. The inherent statistical variation in the process gives definition to anomalous behavior. In the statistical sense, anomalous behavior has an extremely low numerical value for its probability of occurrence. Consequently, if anomalous behavior is not observed, then the process is in control.

There are several applications for control charts with a division between two basic types, i.e., those with attribute data and those with variable data. An attribute is a characteristic that is either present, or is not. Conversely, a variable characteristic is measurable on a continuous scale. The control chart method chosen for our application is termed *Individuals* and *Moving Range.* Symbolically, it is shown as XmR, where X represents individuals, and mR is the moving range. This method can be used for both attribute and variable data. It is the appropriate control chart choice when only a single data point is available per sampling. The Individuals and Moving Range method fits our application because CPI and SPI are variables, and there is just one data point per month for each.

Figure 2 illustrates the XmR control chart. The symbol X used in the figure represents CPI^{-1} or SPI^{-1}. The symbol mR represents the difference in X's value between i = n and i = n+1. The equations, along with the value of the correction factors necessary for creating the chart are also included in the figure. The correction factors (d_2, D_3, and D_4) are derived from statistical theory, and are used to calculate

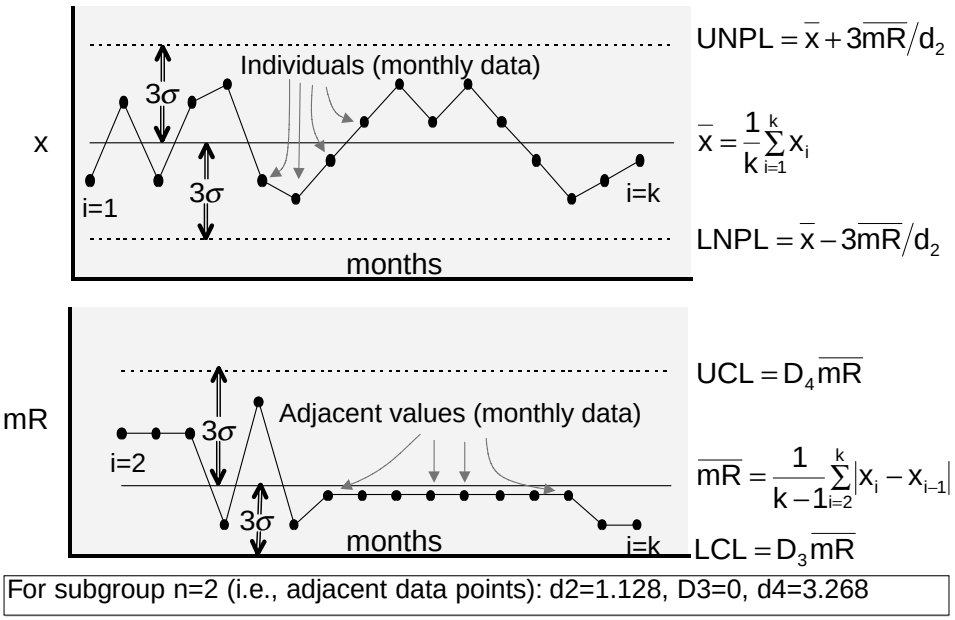


Figure 2. *Control Chart, Individuals and Moving Range*

the control limits of the process (i.e thresholds beyond which a measurement has an extremely low probability of occurrence).

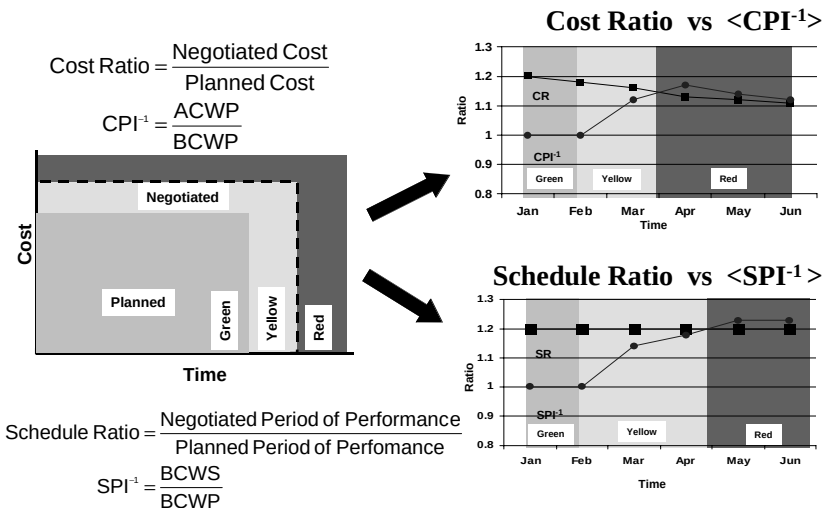
Control limits (upper and lower natural process limits [UNPL, LNPL] and upper and lower control limits [UCL, LCL]) are established at six sigma (6 σ), where sigma is a standard statistical measure of the variation in the process being observed. Measured values outside of the six sigma limits have a probability of occurrence of only 0.27 percent—virtually zero. any measured value occurring outside of these limits is an anomaly requiring management attention.

One of the applications of control limits is testing the capability of the process. For a product to be satisfactory for a customer, it will normally have a specification that establishes its acceptability in terms of upper and lower limits. By comparing the product limits to the process limits, we can predict the percentage of product not expected to meet customer requirements. The calculated value is the probability or risk of manufacturing unacceptable product.

Performance Analysis

As mentioned earlier, we have merged EV and SPC to create another software management tool [3] depicted in Figure 3. By establishing performance limits in the form of cost and schedule ratios, the average of the monthly values for CPI^{-1} and SPI^{-1} can be compared, respectively. The ratios are simply the quotients of the customer requirement to the expected project performance. The cost and schedule ratios establish the poorest perform-

Figure 3. *Performance Analysis*



ance efficiency allowed for the project to achieve the customer requirement.

It is a simple matter to color code the performance. If performance indicates the project will be completed within the planned cost and schedule, the status is green. In this case, if the average value of CPI^{-1} and SPI^{-1} (subsequently shown as $<CPI^{-1}>$ and $<SPI^{-1}>$) is reported to be 1.0 or less, then we can expect the plan to be achieved. If performance indicates the project plan will be exceeded, yet the customer requirement will be met, then the status is yellow. For this case, the values of $<CPI^{-1}>$ and $<SPI^{-1}>$ will exceed 1.0, but be less than the comparable ratio. Of course if the indicators show efficiencies in excess of the ratios (i.e., the limit above which performance is unacceptably poor), then the status is red.

Likewise from the article referenced earlier in this section [3], the color coding of the performance status leads to a recommended management action, i.e., adjustment of overtime or staffing, realignment of personnel, or customer negotiation. The calculation methods for adjusting overtime and staffing are also discussed in the article.

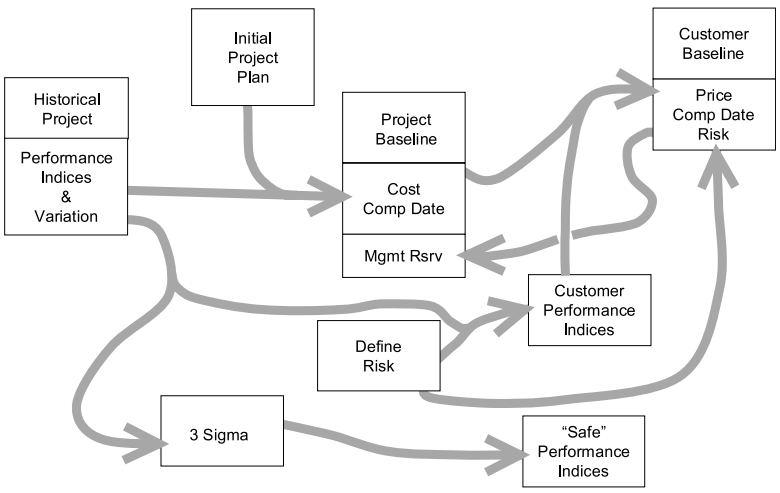
Project Planning

We have used the EV-SPC method for approximately one year as a tool for managing software developments. It occurred to us that the method could also assist with project planning. The basic idea is that data ($<CPI^{-1}>$, $<mR_c>$, $<SPI^{-1}>$, and $<mR_s>$) from past projects could be used to amend the draft project plan to establish a project baseline (cost and completion date). Using the project baseline and risk level that the company or project manager is willing to accept, computations can be made of the customer price (without profit) and delivery completion. The differences between customer values and project baseline then determine the values for management reserve. The planning process is schematically shown in Figure 4.

In the figure the computation of *safe* performance indexes is shown, i.e., the highest average values of CPI^{-1} or SPI^{-1} that will not exceed the 3 sigma variation from the historical average value. There are times when to be safe is not much more expensive in cost and schedule than for the risk defined. This condition occurs when the process is highly refined and is indicated by extremely small values of <mR>. Certainly, if being safe does not make the project's bid noncompetitive, it is the ideal project plan. However, in general all project managers will need to accept some risk of failure to win the contract. This planning method quantifies the risk and computes the commensurate management reserve.

Before elaborating on the computations, the expected result of this planning method needs to be discussed. Our hypothesis is that, at minimum, we could expect the new project to execute closer to the plan. After all, the adjustment to the project baseline is, in essence, the use of a lesson learned. Also, if little change is made to the planning methods, then we could expect the variation of the performance indexes from the new project to be approximately the same as those from the historical project. On the other hand, if improvements are made in the planning method or the software development process, it becomes more difficult to predict the result. Nevertheless, what we hope for is performance indexes closer to the planned values and decreased variation.

Figure 4. *Planning Method*



Calculation Procedure

This procedure follows the flow depicted in Figure 4. It may be helpful to consult the figure as the calculations are described.

Step 1. Select the historical project that has the greatest amount of similarity to the project being planned. The variables of programming language used, staff experience, software engineering environment etc., variation in the work breakdown structure (WBS), and values allocated to the tasks are to be considered in selecting the historical project. Also, the planning team's biases should be considered for historical and new projects, along with the customer's behavior attributes. Once the selection is made, obtain the historical data: $<CPI^{-1}>_h$, $<mR_c>_h$, $<SPI^{-1}>_h$, $<mR_s>_h$. Note that the performance of tasks, which are out of scope with respect to the contract, may be imbedded in these numbers. If the value of these tasks is known, it is appropriate to remove their effects and adjust the historical data accordingly.

Step 2. Develop the draft plan to establish the initial budget at completion (BACi) and period of performance (POPi).

Step 3. Create the baseline project plan by using the performance efficiencies from historical project data to adjust the initial plan.

$POP_i \times <SPI^{-1}>_h = POP_p$

[Start Date + POP_p => Expected Completion Date (CD_p)]

$BAC_i \times <CPI^{-1}>_h = BAC_p$

At this point, it is probably worthwhile for the planning team to reflect again on the differences between the historical and new projects. To finalize the baseline plan, Steps 2 and 3 may need to be iterated.

Step 4. Next, calculate the safe performance indexes for future reference. Recall from the earlier discussion that it may not cost much to be safe. Later on a comparison will be made between safe vs. risk performance.

$<CPI^{-1}>_{3[\sigma]} = 1.0 + 3 <mR_c>/1.128$

$<SPI^{-1}>_{3[\sigma]} = 1.0 + 3 <mR_s>/1.128$

The formulas used above are shown in Figure 2.

Step 5. Define the acceptable risk level. Defining risk is a tricky business. The project manager desires sufficient management reserve to cover all anticipated risk. But, the company wants to win the contract, which means accepting more risk. Therefore, the planning team feels pressure to lower the price and shorten the schedule. The level of risk selected will be a compromise between these two extremes. Generally, the risk associated with achieving the project cost is lower than the risk for delivering the product on time. The planning method allows for establishing risk levels for both cost and schedule. For the remainder of the discussion, the risk acceptable is equated to the probability of failure (P_f).

Step 6. Establish the boundary for red performance. The upper specification limits (USL_c and USL_s) for $<CPI^{-1}>$ and $<SPI^{-1}>$, respectively, are calculated from statistical data. The procedure is general to both cost and schedule; the calculations for each are performed identically. To illustrate this, an example of the calculations for 30 percent risk is shown below.

The probability of failure can be determined from the mathematical table of the cumulative normal distribution [8]:

$$P_f(z) = 1 - F(z), \text{ where } F(z) = \int_{-\infty}^{z} \frac{1}{\sqrt{2\delta}} e^{-z^2/2} dz$$

(The representation of F(z) shown is one of the forms available in mathematical tables; there are other forms that can be used equally as well.)

For risk = 30 percent, $P_f(z) = 0.30 = 1 - F(z)$
Thus, $F(z) = 0.70$

Using the mathematical table of the cumulative normal distribution, identify F(z) values adjacent to F(z) = 0.70.

$F(.52) = 0.6985$, $F(.53) = 0.7019$

Now, estimate the value of z for 30 percent risk (z (@30 percent)) by interpolation.

$$z\,(@30\%) = 0.52 + \left(\frac{0.7000 - 0.6985}{0.7019 - 0.6985}\right)(0.53 - 0.52)$$

Thus, z (@30%) = 0.5244

Using the z equation [6], calculate the value of the USL.

$z = (x - u) / [\sigma]$, where x is the value of the point (in this example, USL), u is the average value (in this example, 1.0), and $[\sigma]$ is the standard deviation (in this example, <mR>/1.128)

Therefore, z (@30%) = (USL – 1.0) / (<mR>/1.128)

With some algebraic manipulation, the equation can be solved for USL.

To illustrate the calculation, we will use a value from an actual project, <mR> = 0.2652.

USL (@30%) = (0.5244) (0.2652)/1.128) + 1.0

Performing the math, the performance index representing the red boundary for 30 percent risk is computed to be 1.1233 for our example.

Step 7. Establish the customer baseline. Knowing the value of USL, the customer baseline can be computed.

$POP_p \times USL_s = POP_c$

Start Date + POP_c = Customer Completion Date (CD_c)

$BAC_p \times USL_c$ = Price (without profit)

The above calculation should be repeated using the appropriate safe index from Step 4 in place of its respective USL multiplier. As stated earlier, to be safe may not raise the price or increase the schedule significantly if the development process is very refined (the average value of <mR> is small).

Step 8. Calculate the management reserve. The management reserve is simply the differences between the customer and project baselines.

For Schedule Reserve, $MR_s = POP_c - POP_p$ (workdays)
or
$MR_s = CD_c - CD_p$ (calendar days)
For Cost Reserve, MR_c = Price – BAC_p (dollars)

Prototype Application

The planning method and calculation procedures were tested against both a historical and current project. The in-work project was planned using knowledge gained from the performance of the historical project. For reference, the products from the historical project were 52 test program sets (TPS). TPSs are a combination of the software and hardware needed to test the performance and diagnose the failures of electronic circuit cards. The project ran for five years and had a peak staffing of 12 engineers. The in-work project was to develop nine TPSs. It has eight engineers presently assigned, and is 29 percent complete.

As we discussed in the calculation procedure, better results are expected when data is used from historical projects having similar attributes to the project to be planned. In this case, the projects are highly similar. The circuit cards requiring the TPSs are similar in structure, application, and component technology. The automated test system, which uses the TPSs for performing the circuit card testing, is the same for both projects. The programming language used for developing the software is also identical for both. The specification defining the TPSs came from the same customer; there are only minor differences. The planning team, project leader, and three engineers are common to both projects. The planning team used the WBS from the historical project with only minor changes. Schedules for the two projects overlapped for approximately nine months.

The historical project has the following performance values: $<CPI^{-1}> = 1.12$, $<mR_c> = 0.40$, $<SPI^{-1}> = 1.18$, $<mR_s> = 0.46$. The numbers were derived from 61 monthly data points. The $<CPI^{-1}>$ and $<SPI^{-1}>$ values indicate poorer than expected performance. While the project had enough cost reserve to avoid an overrun, it did not plan for schedule reserve and did experience problems completing the product deliveries on time. The amount of variation is somewhat larger than the values of <mR> from our other software development areas. Thus, it was thought that better planning was needed.

The results of using the EV-SPC planning method are shown in Table 1. Here you see that the planning team covered a greater amount of schedule risk (35 percent probability of failure) than was accommodated for cost (40 percent probability of failure). To mitigate the anticipated risks, the cost reserve is 8.8 percent of the expected project cost, and the schedule reserve is 15.6 percent of the planned period of performance. As a point of emphasis, the methods described provide a significantly more

Project Plan			
Cost	$1,732,832	Completion Date	January 1, 2002
Cost Reserve	$153,444	Schedule Reserve	4.8 months
Risk	40.2%	Risk	35.2%

Customer			
Price	$1,886,276	Completion Date	April 30, 2002

Table 1. *Method – Results*

complete understanding of the probability of project failure. By having the risk strategy quantified, we can logically expect improved business practice and software project management.

The performance results taken from the new project bear out the hypothesis made earlier. These results are presented in Table 2. As predicted the inverse performance index values are closer to the expected values of 1.0 than for the historical project; cost is closer by 41.7 percent while schedule is closer by 61.1 percent. However, the statistical variation seen in the new project is considerably smaller, something we did not expect. Cost variation is reduced by 47.5 percent, and schedule variation is down by 26.1 percent.

Table 2. *Performance*

	Historical	Prototype
$<CPI^{-1}>$	1.12	0.93
$<mR_C>$	0.40	0.21
$<SPI^{-1}>$	1.18	0.93
$<mR_S>$	0.46	0.34
n	61	10
% Compl	100	28.7

Reflecting on the two projects, adjustments were made to the earned values that were assigned the various WBS tasks during the new project planning. The expectation of improving the project planning is the removal of statistical variation from some of the common cause [6, 7] entities. But overall, the reduction is enhanced from other lessons learned by the planning team, project leader, and the three experienced engineers. They used those lessons from the historical project to guide themselves, and especially the new members of the project team. Thus far many pitfalls experienced during the historical project have been avoided. Significant process improvement is evident.

Summary

The software project planning method proposed in this article incorporates the use of past project performance data, earned value management, and statistical process control. The method provides several outputs:

- Project cost and customer price.
- Expected and customer completion dates.
- Management reserve for both cost and schedule.
- Quantified risk for cost and schedule.
- Statistical quantification of process improvement.

We have shown in the example application that the method may have merit. In general, the results predicted were observed. The new project is performing closer to the plan with less variation. Certainly this is improved software project performance. And with improved performance, it is expected that customers will be increasingly satisfied. Of course the bottom line to achieving customer satisfaction is gaining additional business. We believe the software management tool presented in this article can help to achieve these goals.◆

References

1. Gordon, Creaghe, Risk Analysis and Cost Management, 16th Annual College of Performance Management Conference, May 2000.
2. Sen, Surhita, EVM Concepts in SPC, 16th Annual College of Performance Management Conference, May 2000.
3. Lipke, Walt and Vaughn, Jeff, Statistical Process Control Meets Earned Value, CROSSTALK, June 2000.
4. Lipke, Walter H., Applying Management Reserve to Software Project Management, CROSSTALK, March 1999.
5. Fleming, Quentin W., *Cost/Schedule Control Systems Criteria, The Management Guide to C/SCSC*, Probus, Chicago, 1988.
6. Pitt, Hy, *SPC for the Rest of Us*, Addison-Wesley, Reading, Mass., 1995.
7. Florac, William A. and Carleton, Anita D., *Measuring the Software Process*, Addison-Wesley, Reading, Mass., 1999.
8. Crow, Edwin L., Davis, Francis A., and Maxfield, Margaret W., *Statistics Manual*, Dover Publications, New York, N.Y., 1960.

About the Authors

Walt Lipke is deputy chief of the Software Division at the Oklahoma City Air Logistics Center. The division employs approximately 600 people, primarily, electronics engineers. He has 30 years of experience in the development, maintenance, and management of software for automated testing of avionics. In 1993 with his guidance, the Test Program Set and Industrial Automation (TPS and IA) functions of the division became the first Air Force activity to achieve Level 2 of the Software Engineering Institute Capability Maturity Model® (SEI CMM). In 1996, these functions became the first software activity in federal service to achieve SEI CMM Level 4 distinction. The TPS and IA functions, under his direction, became ISO 9001/TickIT registered in 1998. These same functions were honored in 1999 with the IEEE Computer Society Award for Software Process Achievement. Lipke is a professional engineer with a master's degree in physics.

Mike Jennings is chief of the Avionics Test Software Section, an organization within the Software Division at the Oklahoma City Air Logistics Center. His section develops and maintains software for the automated testing of avionics systems from several weapon systems. He has more than 10 years of automated testing experience in various software development and leadership positions. Jennings earned a bachelor's degree in electrical engineering from Oklahoma State University.

OC-ALC/LAS
Tinker AFB, Okla. 73145-9144
Voice: 405-736-3341
Fax: 405-736-3345
E-mail: Walter.Lipke@tinker.af.mil
Michael.Jennings@tinker.af.mil

Leverage an Estimating Model to Climb the CMM Ladder

Arlene F. Minkiewicz
PRICE Systems L.L.C.

How much value will a software-estimating model add to your organization's efforts to increase software process maturity? A lot, because not only does it aid in tackling the project management aspects of process maturity, but it also aids in progressively sophisticated ways as your organization's maturity increases. A software-estimating tool can provide a language and framework for making valid and useful comparisons at all levels of the organization. This article indicates how your organization's climb up the CMM® ladder can benefit by incorporating a software-estimating tool.

I am frequently asked how a commercial software model will add value to an organization's attempts to increase software process maturity. I like that question because there is a pretty simple answer up front. There are also many details that make for a rich discussion about software process improvement and software cost models. The simple answer is that project management is a very important part of any solid process improvement effort. Being able to perform accurate size, cost, and schedule estimates is vital to good project management.

The more interesting answer requires some digression into the complicated world of process improvement. I like to frame this discussion in the context of Carnegie Mellon University's Software Engineering Institute (SEI) Capability Maturity Model (CMM) for software. I do this not because it is the only vehicle for software process improvement. It is, however, an effort that encapsulates years of research into the software related processes that must be in place and properly understood and executed in order to have a world class software development organization. I will talk briefly about software process maturity and the CMM—what it is and why it is beneficial. This will be followed by a discussion of software-estimating tools—what they do and how they work. Then I will delve into some of the specific process areas where a software-estimating model will help your organization achieve process maturity successes.

Software Process Improvement and the CMM

In the late 1980s, the federal government determined that their job of building software systems would be simplified if they could perform quantitative evaluations of the capability of the subcontractors competing to build these systems. They shared this realization, along with some funding, with SEI and began the project that led to the development of what we call today the CMM for software. Since its introduction in the early 1990s, hundreds of software development organizations have used the CMM not only to assess the maturity of their existing processes, but as a framework to guide their climb to higher levels of process maturity. The CMM has gained tremendous popularity in the industry, so much so that many organizations are finding they must achieve certain levels in order to win software contract awards.

So what is the CMM? It is a model through which a quantitative assessment of an organization's software process maturity can be made. The CMM document for software published by the SEI describes the process areas that should be addressed and gives guidance on the activities required to get those processes in place. This model is based on the premise that real process improvement involves the entire software development organization, not just the groups that build the software. It requires commitment throughout the entire organization. A CMM assessment at an organization results in the assignment of a ranking from one (initial) to five (optimizing) depending on how many of the 18 Key Process Areas (KPAs) have been successfully ingrained in the organizational software development process.

Benefits of Software Process Improvement

Clearly the climb from Level 1 to Level 5 is a long and expensive journey. Why are so many companies willing to do it? There are many ways that process improvement benefits the software development organization—both qualitative and quantitative. Some are fairly hard to measure well. Even those that are easy to measure are often undervalued because it is not until an organization reaches some beginning level of process maturity that the measuring mechanisms are in place. This makes comparisons to the worst case very difficult.

Improved processes result in higher quality products. Product quality is a very hard thing to quantify even though we may count defects per line of code (or other size measure) in the released product. It is still hard to quantify those missed requirements or features that failed to meet any client need. Certainly focus in Level 2 on requirements management and software quality assurance begins to address better analysis and early defect detection—before the product is released rather than after.

However, it is the move to Level 3 that really begins to make an impact on overall product quality. The introduction of software product engineering and intergroup coordination results in products that deliver the right functionality in a low defect package. The introduction of peer reviews starts the process of preventing defects before they begin. This improvement in product quality has the added benefit of lowering maintenance costs.

Improved processes not only result in better products, they lead to better products that can be built in less time for less cost per line of code. There are all kinds of studies that support this, including productivity increases from 60 percent to 100 percent, cycle time decreases from 25 percent to 75 percent, and other numbers all over the map. Yet these numbers need to be viewed in the context of how the studies were done, where the measurements started, and what assumptions were made [1, 2]. Many factors contribute to the increased productivity and reduced cycle time. Processes that focus on forethought, inclusion of all interested parties from the beginning, commitment from all levels of the organization, peer reviews, and training all contribute to working smarter and getting the most out of each hour.

In addition to the dollar-and-cents incentive, process improvement leads to a software development environment

where people are happy to work. Mature organizations offer environments where creativity can thrive within the confines of process. The process areas are meant to constrain the management and execution of the projects, not the content. The mature organization is proficient at predicting cost and schedule, so project plans are realistic. It is also proactive rather than reactive, so developers spend their time writing excellent code instead of putting out fires.

Software-Estimating Benefits

Long before software process improvement and the CMM were common vocabulary, there was wide spread recognition that software project managers needed better ways to estimate the costs and schedules of software development projects. In the early 1970s two concurrent research efforts produced two parametric software cost-estimating models for use in the software development community: Constructive Cost Model (COCOMO) and PRICE Software Model. Since then many new models have been developed as derivatives or expansions of the original offerings. However each new model has evolved differently with individual benefits and shortcomings. The following discussion describes in general terms what a software cost estimating tool does, and how it works without delving into specific details.

Software cost estimating tools require users to input a description of their software project. At a minimum, the cost estimating tools ask the user to describe:

- The size of the software either in source lines of code, function points, or some other sizing metric.
- The anticipated amount of reuse.
- The type of software being developed, including real time, operating systems, Web development, IS, etc.
- The software operating platform, including commercial, military, ground, air, space, or desktop.
- A quantification of the organization's software development productivity.

The tool then derives a cost estimate, and in most cases a schedule estimate, for the project. Processes driving inputs to outputs are: cost estimating relationships derived from regression of actual data, analogies comparing input parameters to existing knowledge bases, algorithms derived from theoretical research, or some combination of these methodologies.

Most tools offer tables, wizards, or knowledge bases to help novice users select the proper inputs or move into new products, platforms, or technologies. The tools also require input about the software development environment (programming language, tools, etc.) and the software development experience of the team. This information is then used to determine the cost and schedule estimates.

While cost and schedule estimates are the main deliverables, there are many other organizational needs the right estimating tool can address. Software project planning is really a balancing act between cost, schedule, quality, and content. The right software-estimating tool can help optimize this balance. Many tools have the capability to estimate latent defects in the delivered product, then use this information to predict maintenance costs. With this knowledge a project manager can make tradeoffs based on the total ownership cost, rather than just development costs.

Most tools have other trade-off and analysis features as well—allowing the user to set a baseline and vary parameters to optimize cost and schedule. Your organization can use a software cost estimating tool to help derive a common language (the tools input parameter set) to describe and compare software development projects, and a common productivity measurement to make reasonable comparisons between projects that have technical and operational differences.

Another important feature that most cost estimating tools deliver is the ability to perform a risk analysis on the cost and schedule estimates so those estimates can be accompanied by a confidence level. They do this by asking the user to specify uncertainty of one or more input parameters, either with a low, high, and most likely value, or across a distribution (Beta, Normal, etc.). The tools then use the specified input uncertainties to replace the point estimate with an output distribution through a simulation technique such as Monte Carlo. From this distribution the estimator can see the likelihood of achieving a particular cost or schedule.

There are, of course, limitations to every software cost estimating tool, which are important to understand. Each tool was created and is maintained using a certain set of data and research that does not include all types of projects, platforms, or technologies. It is important to understand the limits of the tool, and know when you are estimating on the edge or outside of these limits. Many organizations find it best to include in their estimating processes, the use of more than one tool or methodology —one for performing an estimate, and another to use as a sanity check. Doing multiple estimates helps ensure that you will not miss a case where one particular tool is weak. It is also very important, no matter what tool or method you use for your software estimates, to understand what the tool is looking for when it asks for particular parameter values. As with everything else, if your input is not well thought out your output will suffer.

Tools Are Critical in Process Improvement

What part will your software-estimating model play in your organization's drive to a higher process maturity level? There are processes at every level that require the standardization of estimating, data collection, quality control, measurement, and analysis. These are all areas where an estimating tool can help add the structure to define processes. The following addresses some specific KPAs that have direct requirements that an estimating tool will help meet.

Level 2 KPAs

One of the goals of the Software Project Planning KPA is, "Software estimates are documented for use in planning and tracking software projects." The software plan is expected to include size estimates for all work products, along with cost and schedule estimates. A software-estimating model with the capability to estimate software size, cost, and schedule provides an excellent tool for institutionalizing these estimating practices.

The main value a software-estimating tool can add at this point in your process improvement venture is the ability to estimate software cost and schedule consistently and logically. The software-estimating model acts as the lowest common denominator, aiding the process of putting software projects into a com-

mon framework so that information can be learned from each project and applied to many others.

Imagine that you are a software project manager who has just delivered a software project late and over budget. You are gearing up for the next project and would like to learn from this past experience. How do you evaluate where you went wrong? Using your software-estimating tool as a guide, you can determine values for the input parameters that would have led you to the right answer based on the actual experiential data you have collected. Once you have done this, you can, based on what you know about this new project, determine which of these input parameters will need to change and how the input values might change. Using a commercial model for estimating facilitates the creation of a common language for discussing project cost drivers and aids in the development and implementation of a documented process.

One of the goals of software project tracking and oversight is taking and managing corrective actions when the results deviate from the plan. A software-estimating model can be a useful tool in reaching this goal as well. When a well thought-out estimate turns out to be incorrect, this is generally because assumptions made about the project were incorrect. Incorrect assumptions lead to incorrect inputs. The software project team can review the original inputs to the model and compare them to actual information to date. This review helps highlight where incorrect assumptions have been made and offers the opportunity to re-plan the remaining portion of the project based on more accurate versions of assumptions.

Suppose you are involved in an effort to estimate a software project that represents a new market for your organization. In developing your original estimate you made and documented (through an input wizard in the tool) assumptions about your software development team's capability to learn and apply domain knowledge for this new market. You also assumed that the team was proficient in the technologies employed. The critical design review shows the project late and over budget. You are tasked with determining what went wrong, and what the real cost and schedule should be.

You look back at the original estimate and compare it to what you know about the project. First of all, the team took much longer than expected to obtain domain knowledge. Additionally, once the project got underway, they determined that the incorporation of a new development technology was required to achieve all of the project goals. After modifying the inputs to your estimating tool and regenerating the estimate, leaving all other parameters the same, the critical design review cost and schedule is much closer to the actual. You now have an estimate to complete with a new level of credibility. Care should be taken to make sure that use of an estimating tool does not lead you to overlook factors outside the scope of the tool. You may find that there was nothing wrong with your original estimate, but that the missed deadline was due to unforeseeable organizational changes for which no tool could account.

Another process area where a software-estimating tool can be included is the Software Subcontract Management KPA. Not only does there need to be a process for planning software developed on site, but also for reviewing and checking the plans provided by a subcontractor. These plans, too, are based on size, cost, and schedule estimates. Subcontracting organizations can also use the language that a commercial tool offers for talking about things that drive cost.

Level 3 KPAs

As an organization begins to attack the required processes to move from repeatable to defined processes, the focus shifts from the project level to the organizational level. One goal of the Organization Process Focus KPA is coordination of process activities at the organizational level. A key part in achieving this goal is the organization's software process database, which collects process and project data. A commercial estimating model requires inputs to develop a generic framework that describes product and project characteristics (such as functionality, quality, size, complexity, and reuse) in such a way that permits comparisons across the organization. Various organizational groups can use this language to compare dissimilar projects in ways that provide useful analysis to feed the improvement process.

For example, imagine you work for an organization that develops avionics for both military and commercial applications. The military side of the house is attempting to evaluate the cost impact of incorporating a new technology that the commercial side has been using for some time. Using the data learned from the commercial avionics development, the software estimator only needs to change information directly related to the operating platform in applying lessons learned to evaluate how this new technology would change costs on a military application. The input parameters for the model help focus the evaluation on what is different and what is the same when performing this type of comparison, and help remove noise from the comparison.

As processes begin to be defined at an organizational level, the real power of a commercial model becomes clear. Part of the Integrated Software Management KPA requires that an organization use data in the software project database for software planning and estimating. This is a process that cost estimating professionals call calibration, and it is automated as part of the many good estimating tools. As noted earlier, in implementing a commercial tool, the organization has already committed to storing data in the process database in a way that makes comparisons possible at an organizational level. The tool then has the capability to look at this historical data, which describes the actual past performance of the organization, and use it to improve the estimates made from that point.

Another question frequently asked is how inputs to the estimating model might need to change as the company progresses to higher levels of maturity. The beauty of a process improvement approach such as that dictated by the CMM is that once an organization has reached a new level of maturity, they already know the answer to this question. The increased data collection and analysis with more mature organizations provides the information required for calibrating and tailoring the inputs to reflect organizational maturity accurately. The value of the tool itself is improved tremendously by the processes that utilize it.

Level 4 and Beyond

Most software-estimating models contain features that can help meet process needs for the Software Quality Management

KPA. A model that contains a submodel to estimate defects per size measure provides the basis for a process that allows for the control and management of a quality plan through trade-off analyses between quality, schedule, and content goals. This submodel can be calibrated by using organizational data from the process database. Suppose you have as an organizational goal to reduce latent defects in your delivered software by 20 percent. At Level 4 you have enough data in the software process database to evaluate defects delivered in the past and calibrate the estimating tool's defect estimate. Once this is done, you can evaluate every proposed project for estimated defects and make corrections to the project plan. This makes possible the goal to extend the development schedule or reduce the amount of content expected in the given time frame until the quality goal is met.

The Technology Change Management KPA requires that an organization have a well-kept process database with productivity and quality metrics from past projects, and a language and framework for introducing new technology parameters. Your commercial software-estimating model can be an important component in that framework. Its parameters for characterizing software projects constitute an important part of the language. A mature organization has much of the right data about past projects. The cost estimating relationships or knowledge bases in your tool have encapsulated the impacts of new and emerging technologies. The marriage of these two stores of information offers an organization excellent insight into how their existing capabilities and experiences merge with new target technologies, and provides information vital to making the right technology decisions.

Conclusion

Process improvement is a worthwhile investment for any software development organization. The CMM is an excellent resource for any organization planning to improve their processes regardless of the level of process improvement needs. Review of the CMM and concentration in those areas that will meet organizational goals is a must for any organization seriously considering process improvement. The CMM is thorough, comprehensive, and encapsulates a lot of good ideas from many experts in the software process field.

No matter how large or small your process improvement plans are you will find that a commercial software-estimating tool will ease your efforts for standardization and control. The most value will be obtained if you incorporate it into organizational practices, and make it a common language for discussions surrounding things that drive your software costs. It is not until all the projects in the organization can be discussed in a common context that true organizational needs and concerns can be addressed. A commercial model will help provide the framework for these discussions. Not only does it offer a way to make this happen, but the tool you choose adds value to your software process improvement program. Conversely, your software process improvement results will make your tool more valuable to your organization.◆

References

1. McGarry, Frank et al., *Software Process Improvement in the NASA Software Engineering Laboratoty*, Technical Report. CMU/SEI-94-TR-22, December 1994.
2. Oldham, Leon G., et.al., Benefits Realized from Climbing the CMM Ladder, CROSSTALK, May 1999, Vol. 12, No. 5.

Additional Readings

- Craig, Rushby, Software Quality Assurance in a CMM Level 5 Organization, CROSSTALK, May 1999, Vol. 12, No. 5.
- Paulk, Mark C., Practices of High Maturity Organizations, *Proceedings of 1999 SEPG Conference*, Atlanta, Ga., March 1999.
- Paulk, Mark C., *Effective CMM-Based Process Improvement*, Software Engineering Institute, Carnegie Mellon University, 1996.
- Lawlis, Patricia K., A Correlational Study of the CMM and Software Development Performance, CROSSTALK, September 1995, Vol. 8 No. 9.

About the Author

Arlene Minkiewicz is chief scientist at PRICE Systems L.L.C. She leads cost research initiatives for the entire suite of PRICE cost estimating products. Previously, she functioned as lead of the Product Enhancement Team with responsibility for the maintenance and enhancement of all PRICE products. She speaks frequently on software measurement and estimating at conferences, and has published articles in *Software Development*, *ITMS*, and the *British Software Review*.

700 East Gate Drive, Suite 200
Mount Laurel, N.J. 08054
Voice: 856-608-7222
Fax: 856-608-7247
E-mail: arlene.minkiewicz@pricesystems.com
Internet: www.pricesystems.com

Assessing Software Risk

Louis A. Poulin
GRafP Technologies Inc.

This article describes the application of hazard evaluation and prevention to software risk management. This approach has been used by organizations involved in developing information technology (IT) applications in order to assess the probability that serious problems will occur, such as cost overruns, schedule slippage, and products or services that do not satisfy their intended needs.

Prevent Losses, Maximize Opportunities

Most risks arise from dealing with change. Any change translates into a loss or an opportunity, each requiring a decision that in turn will generate more changes, as shown in Figure 1. This can lead to an avalanche effect (usually destructive), or it can be channeled to have a constructive outcome.

In any situation at least three main axes should be considered when attempting to prevent losses for a given entity: the human resources associated with this entity; the tools, equipment, and technology used in (or by) this entity; and the mission that this entity is pursuing (or the function it is performing). Assume that five changes can occur along each axis over a given period of time. Each change offers either a loss or an opportunity. If there are two ways to prevent the loss from occurring and two ways to take advantage of the opportunity, any one change requires examining a total of $C_{90}^{2} = 4{,}005$ relationships before making an optimal decision. Given that at times several changes may occur over a period of one day, this is not an easy task, as depicted in Figure 2, which includes only one change per axis. Napoleon Bonaparte's statement to the effect that all he wanted from his generals was that they be lucky, is therefore not entirely surprising.

Obviously problems that have been anticipated are more likely preventable. Just implementing a few preventive measures perfectly matched to an undesirable event increases the likelihood of preventing losses. Conversely, even a large number of preventive actions are liable to be ineffective if potential problems are poorly anticipated. Crises are bound to occur sooner or later.

For example in the Vietnam conflict, the North Vietnamese were definitely more successful than the French and the United States even though they did not have access to all the material resources that the latter had. Yet they demonstrated great ingenuity at exploiting what they had at their disposal to address the challenges they were facing. It is indeed remarkable that they were able to hold out for so long against two world powers.

It should nevertheless be possible to devise a set of *mechanisms* to monitor undesirable conditions and to prevent problems from occurring. In this way, it becomes possible for an entity to operate at an arbitrarily low likelihood of losses as long as it has the capacity to implement such mechanisms [1].

Application in Software Engineering

Among the fields where the aforementioned principles have been put into practice, assessing risk in IT projects [2] is the one in which we have collected the most information. In these assessments three basic parameters were measured: the risk perception level, the risk mitigation capacity, and the likelihood of problems.

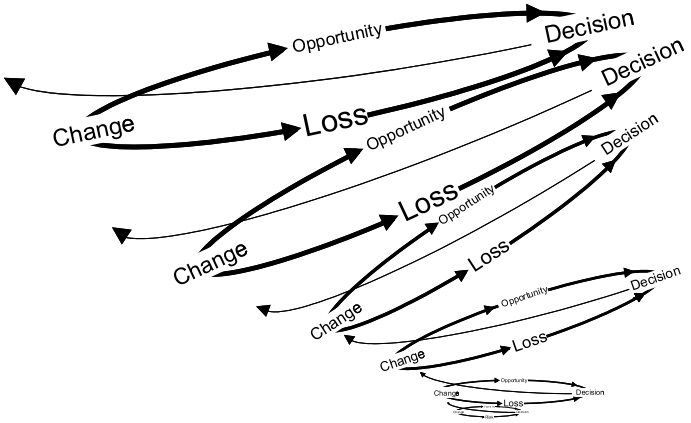


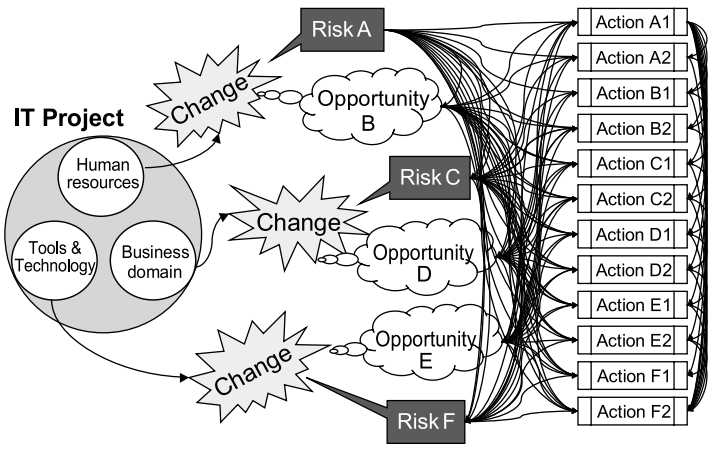
Figure 1. *Loss vs. opportunity*

The Software Engineering Institute's Capability Maturity Model® (Levels 2 and 3) and the Taxonomy-Based Risk Identification [3] were used as the IT assessment framework.

The risk perception level corresponds to the ability to anticipate problems. In an IT context, it is the capacity of professionals assigned to a project to anticipate potential problems and take preventive actions. To some extent, this capacity depends on personnel experience and know-how. It also depends on the risk mitigation capacity because a mature process has a greater capacity to anticipate problems through information provided by its risk mitigation components. In fact, such a process may compensate for a lack of experienced personnel.

In other words, an organization may decide to hire very talented and experienced, high-salaried people to develop the IT application with the help of a minimal and less expensive process. Or it can decide to implement an expensive high-maturity process and hire less experienced, less expensive people. A cost-effective compromise may be to hire a few talented and

Figure 2. *The complexity of preventing losses and exploiting opportunities*



experienced people to develop and implement a high maturity process that captures their know-how and experience, which less experienced people can subsequently apply.

The risk mitigation capacity corresponds to the mechanisms in place to prevent problems. In this context, risk mitigation capacity is similar to process capability, taking into account that some process components have more risk mitigation potential than others.

Finally, the likelihood of problems is the probability that risks will materialize. Again, given the IT framework and defined assessment scope, this represents the probability that serious problems will occur jeopardizing the project or causing failure. Risks in this case include cost overruns, schedule slippages, and products or services that do not satisfy their intended needs.

The experience gained in the course of conducting such assessments has shown that in IT, risks are basically divided into two classes: process-related (common or frequently recurring) risks and project-specific (singular or infrequently recurring) risks. Process-related risks originate from the way methods, tools, procedures, and human resources are integrated to produce a desired outcome. Their nature makes them more prone to recur from project to project (e.g., conditions leading to critical decisions being unduly delayed or taken without having access to all relevant information). Project-specific risks are intimately linked to the nature of a project and are therefore less prone to recur (e.g., conditions leading to a system being unable to handle the volume of information it must process because of network bandwidth limitations).

The breakdown of problems encountered in organizations having undertaken IT projects, based on the information collected so far, is shown in Table 1. This data indicates that process-related risks account for 70 percent of all risks in IT projects where risks of unknown nature are distributed along the ratio of process-related to project-specific risks. In fact, a more accurate statement would be that, on average, risks in IT projects are made up of 30 percent project-specific components and 70 percent process-related components.

Source of Observed Problem (a risk that materialized)	Nature	Relative Freq. of Occurrence
Customers (e.g. poor communication of requirements)	Mostly process-related	15%
System components (e.g. inadequate technical performance)	Mostly project-specific	14%
Development methods (e.g. improper design approach)	Mostly process-related	13%
Management (e.g. critical decision unduly delayed)	Mostly process-related	12%
Suppliers (e.g. inability to deliver as planned)	Mostly process-related	12%
Change management (e.g. incompatible components)	Mostly process-related	10%
Development environment (e.g. unsuitable programming language)	Mostly project-specific	9%
Tests (e.g. incomplete test coverage)	Mostly project-specific	6%
Supervision mechanisms (e.g. irregular tracking of progress)	Mostly process-related	5%
Others	Unknown	4%

Table 1. *Breakdown of problems in IT projects*

Risk Assessment Results Summary

The data collected so far in Europe, South America, and North America using this approach indicates that an IT project has a 33 percent probability of experiencing serious difficulties, including cost overruns, schedule slippage, and products that do not generate anticipated benefits. In terms of frequency, 33 percent of IT projects can expect to experience such problems to the point of failure. This also confirms the finding documented by the Standish Group International showing 31 percent of IT projects are cancelled before completion [4].

Assessment data also showed that, at least in IT, the critical threshold associated with the likelihood of problems appears to be approximately 40 percent. In other words, a project or an organization cannot sustain a likelihood of problems higher than 40 percent for any significant duration relative to the planned or current activities. Consider that a likelihood of problems equal to 50 percent corresponds to operating at random. If such were the case, it would be wishful thinking to expect any successful outcome over a significant period of time.

The same 40 percent value holds true in the financial industry (venture capital), where a portfolio manager will tolerate four investments out of 10 not generating a profit [5]. Anything higher than this ratio will result in restructuring the portfolio in order not to exceed the 40 percent limit, which would result in a certain loss.

The assessment approach has also been used to characterize the Canadian IT industry. Data was collected through 30 comprehensive assessments conducted in Canadian organizations involved in developing products or providing services drawing on IT and software engineering. The size of the assessed organizations ranged from 10 professionals to 250, with an average of 76, and a standard deviation of 64. Table 2 summarizes assessment results.

The results indicate that with a likelihood of problems at 34.3 percent, Canadian organizations can expect to face slightly more difficulties than the average organization, which is characterized by a 33 percent value. Out of the 30 assessed organizations, eight exceeded the aforementioned 40 percent threshold, and in all cases, major difficulties were observed during the 12 to 18 months that followed.

Government organizations have a higher risk mitigation capacity than private industry. But the latter has a higher capac-

Table 2. *Assessment results for all assessed government/private organizations*

	All Organizations		Government Organizations		Private Industry	
Parameter	Average	Standard deviation	Average	Standard deviation	Average	Standard deviation
Risk Mitigation Capacity	60.1%	9.1%	62.9%	11.0%	57.9%	6.9%
Risk Perception Level	38.0%	9.4%	34.0%	10.7%	41.0%	7.2%
Likelihood of Problems	34.3%	11.0%	34.3%	10.9%	34.2%	11.3%

ity of anticipating problems and taking appropriate action. The end result is that both are characterized by the same likelihood of experiencing problems (34.3 percent vs. 34.2 percent).

Reliability of the Approach

A trial was conducted in 1999 with a large government IT project to determine the reliability of the approach. The project called on new technologies and a large pool of resources that did not necessarily share the same processes. It also had particularly challenging coordination aspects stemming from the wide geographical distribution of stakeholders.

On a general level, the trial's main objective was to determine the level of correlation between intrusive (or active) risk assessment techniques such as taking a subsystem apart, conducting audits and inspections vs. non-intrusive (or passive) risk assessment techniques (e.g. keeping a subsystem under remote surveillance or conducting collaborative appraisals). A second objective was to assess the degree of correlation between the measured likelihood of problems when 259 process-embedded risk mitigation mechanisms were investigated versus 404. The number of undesirable situations that are most often encountered in the course of developing or maintaining IT applications was fixed at 163 in both cases. Table 3 summarizes the results of the trial.

The correlation between non-intrusive and intrusive assessment techniques was not as high as expected. In fact, non-intrusive assessments were found to be more accurate than intrusive assessments after the data analysis was completed; intrusive assessments do have an impact on the likelihood of problems because of their disruptive effect on the collected information. On the other hand, non-intrusive assessments do not seem to be overly affected by noise, a concern that had been expressed regarding the use of such techniques. Non-intrusive assessments exhibited a surprisingly high level of correlation considering that two independent teams conducted the field trial. However, while relatively immune from noise, non-intrusive assessment techniques alone tend to be pessimistically biased, whereas intrusive assessment techniques tend to be optimistically biased. Non-intrusive assessments followed by intrusive assessments seem to provide the best results. In this case for an investigation of 404 mitigating mechanisms, the results were: a likelihood of problems of 33.7 percent, a risk mitigation capacity of 61.2 percent, and a risk perception level of 36.7 percent.

	Non-Intrusive Techniques 404 Mitigating Mechanisms	Intrusive Techniques 404 Mitigating Mechanisms	Non-Intrusive Techniques 259 Mitigating Mechanisms
Likelihood of Problems	37.5%	31.0%	37.8%
Risk Mitigation Capacity	58.6%	63.0%	58.0%
Risk Perception Level	36.7%	36.7%	37.3%

Table 3. *Summary of field trial results*

Conclusion

The importance of assessing software risks and of subsequently managing them has slowly been gaining recognition over the last decade. Common sense indeed dictates that reducing the frequency of problems in the course of an IT project will increase the likelihood of a successful delivery.

Through assessments we have conducted during the years, we have observed that the most successful organizations are those that have established sound processes for carrying out their projects while concurrently focusing on anticipating problems and preventing them from occurring. Individuals are known to operate at a constant risk level, and as problems are better anticipated and dealt with, larger projects that present a higher level of risk are initiated, which in turn contribute to the growth of the organization.

It is worth quoting Andrew Grove, CEO of Intel Corp., a company that has had a major impact on IT, in his book *Only the Paranoid Survive*. According to Grove, "Sooner or later, something fundamental will change in your business." The Wallace Corp. is a good example; it won the prestigious Malcom Baldridge Award in 1990 and declared bankruptcy in 1991.◆

References

1. Gallager, Robert G., *Information Theory and Reliable Communication*, John Wiley and Sons, New York, 1968.
2. Poulin, L.A. and Michael Raftus, *Software: Process Risks Identification, Mapping and Evaluation*, Proceedings of the SEI Conference on Risk Management, Virginia Beach, Va., April 1997.
3. Dorofee, Audrey J., Higuera, Ronald P. et al., *Continuous Risk Management Guidebook*, Carnegie Mellon University, Software Engineering Institute, Pittsburgh, Pa. 1996.
4. *Chaos—Application Project and Failure*, The Standish Group International, January 1995.
5. Worzel, Richard, *From Employee to Entrepreneur*, Key Porter Books, Toronto, 1989.

About the Author

Louis A. Poulin assesses the capability of IT organizations and develops hazard evaluation, monitoring, and prevention tools and methodologies. He has a bachelor's in engineering physics, a certificate in Naval engineering, and a master's in electrical engineering. He supervised the development of the tool used to carry out the assessments described in this paper. Previously, Poulin served in the Canadian Navy as a Combat Systems Engineering Officer. He is a member of the Institute of Electrical and Electronics Engineers and a fellow of the Engineering Institute of Canada.

GRafP Technologies Inc.
550 Sherbrooke St. West, Suite 777
Montreal, Canada, H3A 1B9
Voice: 514-847-0900
Fax: 514-847-0400
E-mail: lpoulin@grafp.com
Internet: www.grafp.com

Is Ada Dead or Alive Within the Weapons System World?

Donald Reifer
Reifer Consultants, Inc.

Jeff Craver
U.S. Army

Mike Ellis and Dan Strickland
Dynetics, Inc.

The Theater High Altitude Area Defense program commissioned a study on the long-term viability and staying power of Ada after the demise of its mandate in 1997. The study would help decision makers determine if Ada had the staying power to support future systems, or whether an alternative should be sought. This article puts the results of that study in the public domain.

Since the demise of the Ada mandate on April 29, 1997, little has been said about the debate that raged in the mid-1990s comparing the merits of C++ versus Ada [1] within the weapons system world. Many questions remain such as: "Has the defense community made a wholesale move to competing languages like C++ and technologies like Java?" "Is the migration pattern the same for embedded software as it is for application software?" "Are high quality compilers, bindings, tools and libraries still readily available?" "Is Ada still making converts in academic, industrial and government circles?" "Are vendors making a profit?" "Will vendors exist in the future to satisfy the community's continuing demands for training, compilers, tools and support?" "Will organizations be able to find and hire programmers skilled in Ada language and associated toolsets now and in the future?"

To determine if Ada is still viable, proponents must answer these and many other similar questions.

Like many weapons systems organizations, the Army missile defense community embraced Ada two decades ago because it was the best available alternative to reduce the risk—cost, schedule, and technical—in developing safety critical systems. The language directly supported real-time development needs and risk reduction through standard enforcement. It provided the tools contractors needed to develop highly complex, distributed, real-time systems. If given the chance to make the decision again, under similar circumstances, the Army would still choose Ada.

However, times change along with decision criteria. Today, the missile defense community has millions of lines of Ada software that must be maintained, sustained, and supported up to a 20-year time period. As programs enter full scale development, decision makers cannot help but wonder if Ada will retain the staying power needed for cost effective systems support. Whether to continue using the language or switch to an alternative is a very real question in light of Ada's current status.

To make an informed decision, the THAAD program commissioned a study to address questions relative to the long term viability and staying power of Ada. As part of the study, the program developed a wealth of information that could prove useful to other members of the Department of Defense (DoD) weapons system community relative to Ada's viability. This article shares the information by putting it into the public domain. Hopefully, others will use this information to make informed decisions when answering the question: Is Ada dead or alive?

Viability Assessment

Previous studies have reported that the viability of a programming language is a function of many variables. The accompanying tables 1, 2 and 3 were developed via a Delphi exercise by Reifer Consultants, Inc. (RCI) by having a group of software managers rank items per the criteria listed using a scale of one to five. In these tables Ada scores well in language evaluation when the richness of the language and its degree of standardization are taken into account. Its support for real-time development and reuse features provides facilities that users who work within the weapons systems community always deem important.

Table 1. *Language Evaluation (rating scale 1 to 5 [highest])*

Factors\Language	Ada	C/C++
Core Features • Strong Typing • Exception Handling	5	4
Degree of Standardization	5	4
Object-Oriented Support	5	5
Reuse Facilities	5	4
Real-Time Programming Support	5	3
Subtotal	25	20

Table 2. *Compiler/Tool Availability (rating scale 1 to 5 [highest])*

Factors\Language	Ada	C/C++
Optimizing compilers available for current host/target platforms	5	5
Optimizing compilers planned for future host/target platforms	3	5
Bindings to existing systems software available (POSIX, Windows 98, etc.)	5	5
Bindings to future systems software available (Linux, Windows 2000, etc.)	4	5
Bindings to GUIs and generators available (Fresco, etc.)	4	5
Rich libraries available (run-time, math, class, building blocks, etc.)	4	5
Compiler support tools available (syntax-directed editor, symbolic debugger, etc.)	4	5
Inexpensive visual toolset available	2	5
Subtotal	31	40

Table 3. *Education and Training Support (rating scale 1 to 5 [highest])*

Factors\Language	Ada	C/C++
Popularity	2	5
Public training offerings available	2	5
Literature and textbooks readily available	4	5
Consultants and subcontractors with skills in language available for hire	2	5
Contractor core competency with language and toolset	5	3
Subtotal	15	23

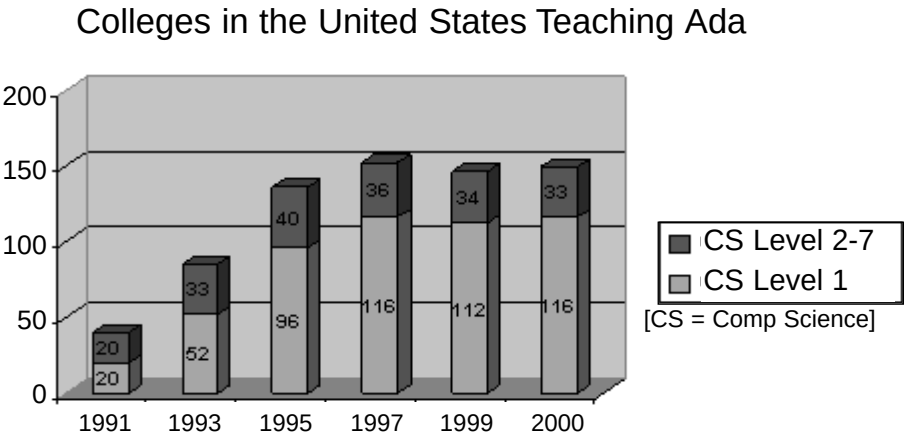
However, its lack of bindings, tools, libraries and inexpensive compilers has been a weakness that has caused users to shy away from selecting Ada in the past, especially for command and control projects. Most importantly, a lack of popularity and education and training shortfalls has detracted from Ada's use in new programs. Today's projects demand a language whose products are stimulated by market forces, not edicts. Such forces stimulate product developments along with language learning and use.

These factors can have a large impact on a program like THAAD. For example, the cost of a compiler for a new target machine is prohibitive if the project, not the market, has to stimulate product development. The cost to produce just a new code generator for THAAD would exceed $1 million.

Additionally, it would take 18 months to field this compiler; and the activity would be on the critical path due to impacts from potential schedule delays. As a result, practical concerns epitomized by the following trends also play an important part in assessing the viability of language alternatives:

• **Vendor/Tool Availability—**Figure 1 summarizes our findings relative to the availability of vendors, compilers and tools. This chart and Figures 2 and 3 were developed using public data available on Ada's Web page (www.adahome.com) by RCI for the THAAD program office. As the figures illustrate, the number of vendors generating Ada products has been cut in half since 1994. Of course, some of the firms that disappeared were acquired. Others just went out of business. In addition, the number of compilers has decreased dramatically as users of Ada 83 have migrated to Ada 95.

On the upside, tools supplied with the compilation systems (debuggers, editors, etc.) that have survived are rich and capable, especially those that are part of a Multi-Language Support System (MLSS). But the cost of these compilers and tools is high compared with the alternatives. In addition, their availability for new platforms is questionable. To ensure options, the program would have to fund the compiler developments and maintenance. As shown, this alternative adds cost to the program, and because compilers are on the critical path, increases programmatic risks.



- Number of colleges teaching Ada peaks in the year (1997) when the government mandate (subsidy) stops.
- Trend from 1997 is relatively flat – academia is slow to change.

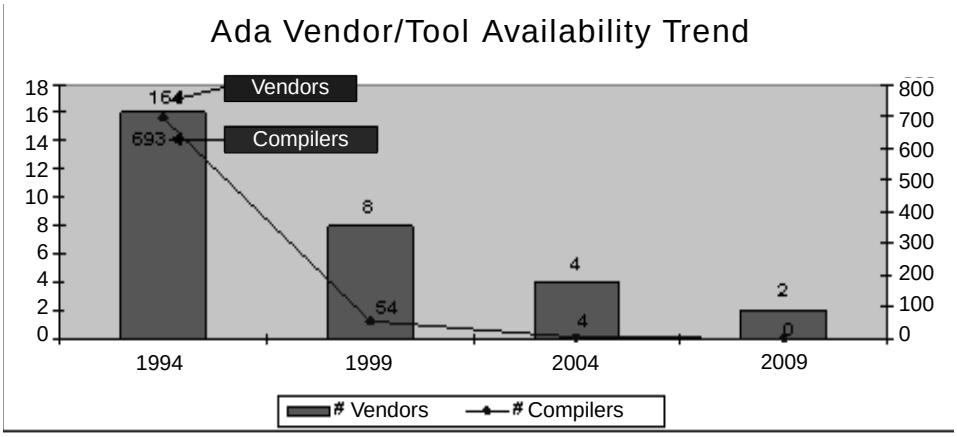
Figure 2 Trends in Academia

• **New Starts/Language Use Trends**—Most vendors interviewed agreed that Ada would continue like Jovial and other DoD programming languages as a niche market. Most of their business was concentrated in maintenance. Very few new projects were selecting Ada. The reasons for this lack of popularity are highlighted in Tables 1 through 3 above. Our primary concern is that without a large market to fuel future compiler and tool developments, firms will rely on projects like THAAD to fund innovations and compilers for new platforms and targets. The costs for this are prohibitive relative to available options.

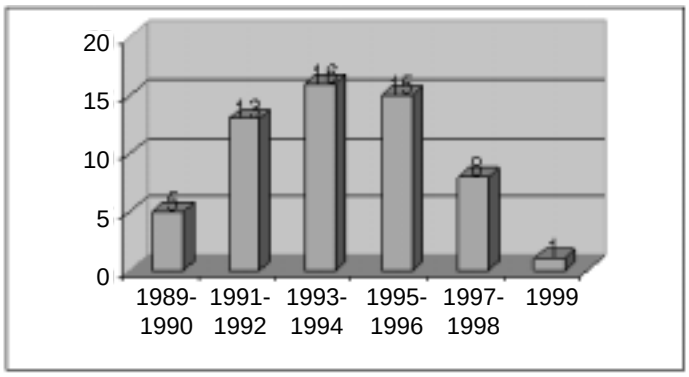
• **European Use Trends**—THAAD kept hearing from Ada advocates that development was stronger in Europe. In response, our team surveyed the overseas marketplace to see if things were any different outside of the United States. This survey confirmed that the European marketplace mirrors the U.S market. Like the United States, there were few new starts for Ada efforts in Europe. Similarly, most Ada efforts identified in the European weapons systems community were focused on maintenance and upgrades. Again, as in the United States, the government Ada supporters were neither funding R&D nor urging their contractors to use the language.

• **Academic Trends**—Figure 2 summarizes our findings relative to Ada's academic trends. As this figure illustrates, Ada use by colleges and universities seems to have peaked in 1997. While Ada is recognized as an excellent teaching lan-

Figure 1. *Tool/Vendor Availability*



- From 1994 to 1999, vendors cut to half (16 to 8), compilers to 8%.
- Continuing trend indicates no pure Ada compilers by 2009.



- 1999 book was not new, but a revision of a 1997 work.
- Trend indicates that interest in Ada is diminishing rapidly.

Figure 3 Publication Trends

guage for introduction to computer science, its use trend from 1997 to present is relatively flat. In addition, the number of Ada seminars offered by firms specializing in educating and training professionals working in the field has fallen off sharply. If these trends continue, it may become increasingly difficult to find programmers skilled in using Ada programming language to staff projects in the near future.

• **Publication Trends**—To indicate popularity, the THAAD team surveyed publications about Ada. Figure 3 summarizes the results, which indicate that interest in Ada is diminishing rapidly. This trend confirms that Ada is increasingly becoming a niche market inherently characterized by high costs, low demand, and lack of innovation; i.e., limited competition forces a degree of complacency.

As a final indicator of popularity, the THAAD team investigated conferences and professional publications. The decline of the Tri-Ada conference, seeming lack of interest for Ada at conferences such as the Software Technology Conference, and diminishing number of sponsors for Ada Letters does not paint a good picture for the future staying power of the language.

Productivity Assessment

Ada supporters would argue that its cost-benefits alleviate these and other concerns. However, the productivity data that is summarized in Table 4 shows that Ada no longer has an edge over other object-oriented languages like C/C++ within the weapons systems domain. RCI developed the information in this table by analyzing cost and productivity data for more than 1,500 projects within their historical databases. The trends indicate that the productivity gap between languages has narrowed, and the competition has caught up with Ada (i.e., see [2] for a 1995 snapshot of the RCI databases).

Table 4. *Cost Per Delivered SLOC by Language/Application Domain*

Application Domain	Ada83	Ada95	C/C++	3GL	Norm
Command and control	70	*	50	100	75
Information systems	25	25	20	35	30
Telecommunications	50	35	40	80	60
Weapons – Airborne	150	125	125	225	175
Weapons – Missile	150	*	*	250	200
Weapons – Spaceborne	150	*	150	200	175
Weapons – Ground	75	75	50	90	75
* Not enough data available					

Conclusions

The results summarized so far should not surprise anyone. Based upon the evaluation factors and identified trends, the verdict on Ada use is in: Lack of DoD institutional support and popularity has weakened its position relative to the competition. Not surprisingly vendors report only one in 10 projects within the weapons system community seem to be selecting Ada for new developments.

The issues involved are pervasive even when you have millions of lines of code to support while looking at developing/upgrading code for new platforms. While the Ada language provides superior support for weapons systems development, the investments needed to continue and sustain its use are large, and for the most part, not budgeted. Neither are the costs needed to convert millions of lines of code from Ada to C/C++.

While the THAAD project supports Ada's continued use, it must do what is in the best interests of the project. However, blanket approval to change to another language is in nobody's best interests. In response, THAAD has elected to permit its contractors to change to another language only when it makes both economic and technical sense. They must justify the conversion technically and in terms of the lifecycle costs before being given permission to change. In addition, they must also develop a transition plan that details how the transition will take place as part of the approval process. Then, projects like THAAD can figure out how to manage the transition and amortize the costs as part of an existing budget.

The approach THAAD has taken is consistent with current DoD policy regarding Ada, which calls for doing what makes sense in the long term for the program. THAAD recommends that other programs critically examine their situation before abandoning Ada because of its technical strengths as a real-time programming language.

THAAD is also investigating using MLSS. Such tool systems permit the vendors to reuse their existing language front-ends (syntax analyzers, etc.) with common back-end tools (code generators, editors, debuggers, etc.). This reduces problems associated with learning different toolsets and increases availability of bindings, tools, and libraries. Yet caution must be exercised to select compilers that enforce and implement published language standards in keeping with required real-time, safety critical systems.

In conclusion, Ada is not dead. It is alive and providing quality support to programs like THAAD. However, its future is not assured. Trends indicate that Ada is following the direction of Jovial and other DoD programming languages. In response, the project must continuously address the risks, and do what makes economic sense for the program. Because others will probably elect to follow suit, we have put the results of our study in the public domain.◆

References

1. U.S. Air Force, *Ada and C++: A Business Case Analysis*, 1991.
2. Donald J. Reifer, Quantifying the Debate: Ada versus C++, CROSSTALK, July 1996.

About the Authors

Donald J. Reifer is a consultant specializing in change management at Reifer Consultants, Inc. in Torrance, Calif. He has more than 30 years of experience managing large software projects and putting software technology to work in Fortune 500 firms. From 1993 to 1995, he was chief of the Ada Joint Project Office, technical advisor to the Center for Software, and chief of the Departmetn of Defense Software Reuse Initiative under an Intergovernmental Personnel Act assignment with the Defense Information Systems Agency. Reifer currently helps clients insert product line and component-based software engineering technologies into their software operations.

Reifer Consultants Inc.
P.O. Box 4046, Torrance, Calif. 90510
E-mail: d.reifer@ieee.org
Voice: 310-530-4493

Jeff Craver is the THAAD System Software Engineering division chief. He has more than eight years experience in software acquisition and development process improvement of Department of Defense systems.

U.S. Army Space & Missile Defense Command
SFAE-AMD-THA-W-SW, P.O. Box 1500
Huntsville, Ala. 35807
E-mail: Jeff.craver@thaad.army.mil
Voice: 256-955-1828

Mike Ellis works for Dynetics Inc. as the System Software Engineering branch chief. He has more than 24 years experience in large system software development, quality assurance, and test.

Dynetics, Inc.
990 Explorer Blvd.
Huntsville, Ala. 35806
E-mail: mike.ellis@dynetics.com
Voice: 256-964-4614

Dan Strickland works for Dynetics Inc. as a software engineer. He has specialized in the software metrics and software cost estimation fields of software engineering.

Dynetics Inc.
990 Explorer Blvd.
Huntsville, Ala. 35806
E-mail: daniel.strickland@dynetics.com
Voice: 256-964-4619

Ed. Note:

Ada has been surrounded by controversy almost since its inception. In this issue we offer one perspective on the current state of Ada and how this affects technology decisions for weapons systems. An upcoming issue will provide an opposing point of view … stay tuned.

Project Management (PM) Web Sites

www.pmforum.org **Project Management World Today**
This is an on-line publication that contains notices, reports, news and information related to project management from around the world. Each issue features editorials and presentations by some of the world's leading project management experts, on leading-edge issues and concepts.

www.ipma.ch **International Project Management Association**
The IPMA is a nonprofit organization based in the U.K. It promotes PM through its global membership network of national PM associations. Currently IPMA comprises 28 national associations representing approximately 20,000 members primarily in Europe but also in Africa and Asia. Moreover, institutional contacts to National Associations in North America, Australia and South Africa ensure a true global dimension to the work of IPMA. The site features listings of conferences, seminars, research, global forum, training courses and quarterly newsletter.

www.infogoal.com/pmc/pmchome.htm **Project Management Center**
This site is dedicated to those with an investment in PM. The site brings information together to cut back on Web cruising and provides practical help. It offers the latest news, articles, software, case studies, links, etc.

www.pmboulevard.com/home.jsp **PMBoulevard**
This PM Community connects you to others facing the same PM challenges you do every day. You can share ideas, compare notes, ask for help, find out the latest news, purchase materials and access other relevant project management sites. There is a calendar of events, a PM Bulletin Board that gives you access to project managers everywhere. Guests can view the ongoing discussion. Members and subscribers can take part in the action by posting messages and replies. It also features a news link, bookstore and PM links.

www.projectzone.com **ProjectZone**
The ProjectZone is a community of technology project leaders discovering, learning, inventing, and teaching each other better ways to lead and manage teams and projects. The material on this site is written by project leaders for project leaders as a volunteer effort, and reflects their experiences and opinions. You will find a variety of different, perhaps even conflicting, points of view here. The site is divided into four distinct zones—strategy, people, structure, and process. Articles on project leadership and management topics can be found in each zone.

www.newgrange.org **NewGrange Center for Project Management**
The NewGrange Center for Project Management was started in early 1997 as a nonprofit professional organization. Its mission is to further the discussion of project management as a professional discipline. The focus is a hands-on, practical approach to project management to determine what really works, why it works, and how to replicate it consistently.

www.allpm.com **The Project Manager's Resource Center**
ALLPM is a site for Information Technology (IT) project management information and resources. It is a clearinghouse of discussion forums, resource links, conference and seminar listings, news releases, contract awards and more.

www.project-manager.com **Project-Manager**
This site is an online guide for anyone who must plan, implement and complete a commercial project. Project-Manager is a learning experience. Here you can get professional advice. Upgrade personal skills. Exchange ideas. Above all, get your project up and running on time and within budget. The site also has links to various Product Depots where you may source equipment, machinery and supplies.

Reaching Level 3 Is Like Traveling a Wide Sea

Paul J. Kimmerly
DFAS Information and Technology Directorate

This article looks at the changes facing an organization as it moves from CMM® Level 2 to Level 3. The author compares the journey to crossing the Bermuda Triangle. Dangers lurk, but there are steps an organization can take to change from the project-centered view of Level 2 to the broader organizational view of Level 3. The article focuses on the action of the organization's management and software engineering process group.

So your organization has reached Capability Maturity Model (CMM) Level 2, and you have decided to move on to Level 3. What does that mean? Where exactly do you go?

The organization faces a difficult journey across uncharted waters to Level 3. The focus of Level 2 is very clear and very pragmatic. Level 3 is a little foggy. Organizations can go adrift as they search for a course from project-focused improvement to organization-focused improvement. At this point the journey to improvement can enter an area like the Bermuda Triangle. There are a number of similarities between the trip from Level 2 to Level 3 and the trip across the wide Sargasso Sea. Organizations' challenges can be compared to three major concerns that plague those entering the triangle: no clear direction, disoriented instrument readings, and motionless waters.

No Clear Direction

Ships entering the Sargasso Sea at the heart of the Bermuda Triangle encounter an area of unusually calm waters. The area contains a large mass of seaweed, or *sargasso,* that adds to the its sluggish nature. Early navigators lost their sense of direction in this unusual area. An organization can fall into this same trap. An organization that has just reached a repeatable level of process maturity run the danger of resting on its laurels. It is easy to enjoy the relative calm of a repeatable level of performance.

Such organizations need a strong captain. Without one, the calm waters can lead the ship to run adrift. A loss of course could lead to falling back to earlier behaviors. The captain that led the ship to the repeatable level must plan a change in course to keep them moving. At Level 2, projects organized their software development practices to a degree of discipline that can repeat past success. This means that each crew understands how to set the rigging and trim the sails. What if you have more than one ship? Will the same practices that work for a sloop work for a schooner? What happens if you add a ship to the fleet?

At this point, the captain must step back and look at the whole fleet. All the ships have similar needs and can benefit from the experience of the others; however, they all have unique needs as well. To address organizational needs, the captain must assemble a command staff that has the responsibility for making sure the individual ships work together. In a software organization, this is the Software Engineering Process Group (SEPG). The SEPG serves as a communication channel from the captain to the commanders of the individual ships. The SEPG also monitors the sailing requirements of each ship to determine how it relates to the others and to find the best practices used by each ship. The SEPG works for the captain while serving as an advocate group that supports every ship commander. In this role, they ensure that the captain's vision is communicated clearly to the ship commanders. The SEPG helps ship commanders put that vision into practice.

The SEPG works with the captain to set the fleet's direction and to define the sailing process to follow that direction. The SEPG works with individual commanders to build a standard fleet-wide process for trimming the sails and battening down the hatches. After the fleet standards have been set, the SEPG goes over the processes with the individual ship commanders to make sure they fit. The schooner may need specific steps to set its many sails. A sloop has a different set of sails and may have different types of hatches. The standard fleet process can be tailored to fit the needs of the individual ships. These standards help the ships determine what to do when they reach unknown waters. When the ships enter calm waters, they all know to take the same steps to sail through them at the fastest possible speed.

When the fleet is outfitted with new vessels, the captain and SEPG meet with the commanders to see how well the standard sailing processes fit. If a steam powered ship is added to the fleet, its needs will be very different; however, some basic fleet processes will still apply. While the new commanders' uniforms are being tailored, so are the fleet's processes.

In a software development organization this comes into play when new projects are developed. If the organization tackles a new project from a new customer, the SEPG works with the new commander to tailor the standard software process and establish a life-cycle flow. When the new commander looks to build the details of his process, he chooses from proven methods used throughout the fleet. He may take the estimating details from Project A and the requirements management process from Project B. If a new technology is being brought into play, the old processes may not fit, and new details will need to be developed.

Without a strong captain and dedicated command staff, an organization faces might run adrift in new territory. The challenges of the Sargasso Sea may prove too much for Level 2 ships, but should be a reasonable challenge to a Level 3 fleet.

Disoriented Instrument Readings

Legendary Air Force Flight 19 lost its way over the Bermuda Triangle. In-flight, the commander radioed that his instruments were acting funny. Sailors since the days of Columbus have also reported problems when crossing the area. Software development organizations rely on instruments, too. Level 2 organizations track size, effort, and schedule, but they may not track them well. The fleet is learning how to use its instruments.

Even Level 1 organizations generate numbers. Some of them may not be collected or reported as consistently as they could be, but the numbers are there. As an organization progresses to Level 2, it gets a better handle on its numbers. It may

be the first time the captain sees some usable numbers. What happens if the numbers look worse as time goes on? What could cause such a problem? How can it be fixed?

Since the fleet may be inexperienced in using instruments for navigation, it may take some wrong turns. When numbers are collected for the first time, there is no historical information to compare to them. Are they good or bad? They may be neither or both—it is just too hard to tell. If the numbers go up or down with the second iteration, are they good or bad? This time, there may be a way to tell.

The key to using the instruments is the reliability of the data. Numbers on things like expended effort and defects may look worse after the ship has sailed to a few new ports. Part of this comes from growing pains of learning. As people become more familiar with capturing defects and more comfortable with how to charge their time, numbers may go up and look worse. Everyone may feel like the ship is sailing smoothly, but the instruments show something else. At this point, the captain must be patient. Realigning the sails or forcing a commander to walk the plank is probably not the answer. Calibrating the instruments is better.

The captain and the SEPG should look for a measurement that is the most consistent. In the software development world, both size and effort lend themselves well to this task. Size measurements like function points or source lines of code can be defined precisely enough so that a size measurement to one ship is the same as a size measurement to the next ship. Having such a normalization factor is key to comparing and combining measures into meaningful metrics. Similarly, if an expended hour is clearly defined for the entire fleet and used consistently by all ships, it can be used as the normalizing measure.

Starting with a normalization factor like function points, an organization can calculate such metrics as defects per 100 function points, hours per function point developed, and cost per function point. Each value should be calculated for each ship for each time it sails. The captain and ship commanders should be aware of such numbers. Each time a trip is made, the numbers can be compared to those calculated for the previous voyage or series of voyages. With some historical data for comparison, the captain and ship commanders can determine the success of the current voyage. They can also predict with some degree of certainty how the next voyage will go. If the voyage is dramatically different from the previous ones, the captain can look for special causes like bad weather or the drag of the sargasso. Even these special causes can be factored in to predict future performance.

In a software organization, each software release represents a voyage. As the measurement process improves and historical data are built, some comparisons can be made to previous releases. If the defect rate per 100 function points goes up, the ship commander may want to look for special causes. If a change is made to the process, the ship commander will want to see the effect on defect rates, delivery rates, and cost factors. Each voyage builds a history and helps set expectations for ship performance. By reviewing current results against those expectations, the commanders gain greater understanding of their process and greater insight into needed changes to the process.

The Level 2 voyage gathered information that can be used for the Level 3 crossing. The information allows the captain and ship commanders to make informed decisions on how to manage the fleet's activities.

Motionless Waters

As stated above, early navigators found unusually calm waters in the Sargasso Sea. The log kept by Columbus mentions several days floundering in the still waters of what is now called the Bermuda Triangle. The crew became restless, and talk of mutiny began. In software development organizations that reach Level 2, such mutiny can come in the form of regression to earlier behaviors. The resulting loss of direction in the motionless waters can also lead to losing sight of what got the fleet to its current position. For the fleet to successfully negotiate these waters, everyone must coordinate their efforts and communicate clearly.

The legend of Flight 19 also shows the problems with faulty communication and coordination. While lost over the Triangle, the flight's commander began having radio problems. Poor visibility made him disoriented, and his failing radio made communication spotty. While members of his flight suggested they were flying along the Atlantic seaboard, he insisted they were in the Gulf of Mexico. When he gave them the order to fly east to look for Florida, he simply sent them further into the Triangle, where they were lost.

What can the captain do to ensure the fleet stays together? How can the ship commanders relate their concerns to the captain? How can the crew members become involved? Should the fleet rely on semaphore flags or Morse code?

For a software development organization, a lot depends on how the organization approached its initial process improvement activities. If the captain's message focuses on the grade, the organization runs a higher risk of reverting to less mature behaviors. The rush for a grade can motivate a crew to put a lot of changes in place. However, it often does not reinforce the desired behaviors that go along with the higher maturity level. After the assessment team has left the ship, crew members fall back on the familiar and disregard the new. If the captain has not communicated intentions beyond the initial improvement efforts, a ship confronting the still waters of inactivity is more susceptible to covert resistance and may court open mutiny.

A software organization's captain must set clear goals based on a well communicated mission and vision. The importance of good communication cannot be stressed too much. The captain and his command staff must constantly work to ensure that the crew understands their role in executing the mission and vision.

The role of the captain and command staff is critical to avoid stagnation or mutiny. As stated earlier, the captain must set a clear course for the fleet as well as communicate that course to the ship commanders. If they do not act accordingly, the fleet remains trapped in Level 2 waters with each ship taking care of its own concerns. The good of the fleet becomes lost in each commander's private concern. The captain must reinforce the fleet's vision and goals. In turn, the commanders must communicate the course to all of their crew members. The entire fleet must be aware of the course for the individual ships to work together. The SEPG plays a critical role in coordinating the fleet's efforts.

Different ships in the fleet must coordinate their activities. By focusing on understanding a ship's needs and finding depend-

able methods of communication, the captain and the SEPG can ensure they consistently convey the right messages to keep the fleet working together. As stated earlier, the SEPG serves as the fleet's channel for communication. While the SEPG does not work for any individual ship, it does work for all of the ships. The SEPG coordinates the improvement activities and ensures that each ship fits in an organization's armada. Regular communication between the SEPG and all parts of the fleet is critical to success. The SEPG serves as the captain's ears when listening to concerns from the commanders. It must also serve as the captain's voice when explaining and coordinating improvement activities. The SEPG also assists commanders in communicating directions to the crew. Additionally, the SEPG must listen to crew member concerns and communicate them back up to the commanders.

The ship commanders must be able to communicate their needs to the captain without fear. While the captain must balance the many concerns of the fleet, the commanders must balance new directives with the fleet's primary mission. The captain must realize that commanders' perceptions drive their behaviors. The clarity and consistency of the message passed down the chain of command shapes how the fleet reacts. The crew listens to commands, but watches actions. The two must be aligned for the fleet to move in the captain's chosen direction.

One way to ensure this is to form a management steering group (MSG) consisting of the captain and his ship commanders with the SEPG in a consulting role. The MSG sets the direction for the organization. Working together to set the organization's mission, vision, and goals, the captain and the ship commanders can communicate a shared message to the organization.

While sailing across Level 2, the ships established their own courses. When facing the expanse of the Level 3 sea, the entire fleet from the captain to the crew must understand the direction of the entire fleet. Through consistent communication and coordination, they can ensure the fleet moves as a whole and finds its way out of the still waters of complacency and routine.

Summary

The Level 3 Triangle holds many challenges for captain and fleet. The stagnant water can make a fleet lose its course. It is critical that the captain be a person of strong mind and clear vision. It is the captain's job to set a direction for all to follow. If each ship sets its own course, the fleet will disperse. Even if instruments seem to be giving false readings, the captain and commanders must trust processes that the fleet has established. That trust will be rewarded when future efforts can be predicted from current ones, and when information can give the leaders a clearer picture of the course being followed. To negotiate the motionless waters, the captain and the SEPG must lead the organization to shift from a ship-centric view to one that encompasses the whole fleet. That view must be passed down clearly from the captain, through the commanders, to each crew member. Crew members, in turn, must be able to communicate concerns and ideas back up the chain of command. These coordinated efforts make it easier for the fleet to find its way out of still waters. By addressing these challenges, the captain can ensure that all the ships set their sails correctly and follow the right tack.

About the Author

Paul Kimmerly has 13 years experience in software development for different incarnations of the Defense Finance and Accounting Service (DFAS) Information and Technology Directorate. Since 1993, he has served as a member of the SEPG , chairing the group for the past five years. In addition to local duties, he chaired a group representing the six software development sites within DFAS that addressed process improvement issues. Paul is a member of the Kansas City SPIN and has given several presentations to the group. He presented part of a tutorial entitled *Transition Success from the Field* at the 1997 SEI Symposium. He also served on a panel at the 2000 SEI Symposium, Statistical Process Control as a Method of Continual Improvement. His article *Quietly Making Noise: A Parrothead's Look at Software Process Improvement* was published in the May 1998 edition of CROSSTALK.

DFAS-FSAKC/KZ
1500 East 95th Street
Kansas City, Mo. 64197
Voice: 816-926-5364 DSN 465-5364
Fax: 816-926-6969 DSN 465-6969
E-mail: pjkimmerly@cleveland.dfas.mil

Coming Events

January 18-19
2001 Measurement Science Conference
www.msc-conf.com/findex.html#cfp2001.html

January 25-27
Ryerson 2001: A Software Approach
www.ryerson.ca/~csie/2001

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

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

March 5-8
Mensch and Computer 2001
http://mc2001.informatik.uni-hamburg.de

March 12-15
SEPG 2001 FOCUSING ON THE DELTA
Software Engineering Process Group Conference
www.sei.cmu.edu/products/events/sepg

March 31-April 5
Conference on Human Factors in Computing Systems
www.acm.org/sigs/sigchi/chi2001

April 22-26
Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies
www.ieee-infocom.org/2001

April 29-May 3
STC
Software Technology Conference (STC 2001)
www.stc-online.org

CrossTalk **Article Index 2000**

The Determining Factor [Open Forum]	D. Dynes	May
Statistical Process Control Meets Earned Value	W. Lipke, J. Vaughn	June
Proven Techniques for Efficiently Generating and Testing SW Programs	K. Wegner	June
Large SW Systems–Back to Basics [Open Forum]	J. Evans	June
The V Model [Web Addition]	Hirschberg M.	June
Requirements Elicitation in Open-Source Programs	L. Henderson	July
Collaborative Software Development	D. Mann	August
New Application of the CONOPS	D. Ammala	August
Is Ada Dead or Alive Within the Weapons System World?	D. Reifer, J. Craver M. Ellis, D Strickland	December

Network Security

Restoring Cyber-Security	B. Crittenton	January
The Systems Security Engineering CMM	R. Hefner, R. Knode, M. Schanken	October
Improving the Security of Networked Systems	J. Allen, C. Alberts, S. Behrens, B. Laswell, W. Wilson	October
The Survivability Imperative: Protecting Critical Systems	R. Linger, R. Ellison, T. Longstaff, N. Mead	October
Avoiding the Trial-by-Fire Approach to Security Incidents	M. West-Brown	October
Security Often Sacrificed for Convenience [Open Forum]	S. Hernan	October
Taming the Cyber-Frontier: Security is Not Enough	P. Toscano	November

Process Improvement

Reducing Software Project Productivity Risk	R. Bechtold	May
Goal-Problem Approach for Scoping an Improvement Program	M. Sakry, N. Potter	May
Four R's of Software Process Improvement	J. Rothman	May
The Demarcation Zone: Surviving A CMM Assessment	D. Jacobs	August
Software Best Practice Development: An Experience	G. Jackelen	August
Software Risk Management: The Practical Approach	G. Holt, K. Phillips	August
Avoid Self-Inflicted Wounds in Applying CMM to ATP and Support	D. Putman	October
The Wide Level 3 Sea [Open Forum]	P. Kimmerly	December

Project Management

What We Have Learned	L. Putnam	June
Project Clarity Through Stakeholder Analysis	L. Smith	December
Project Planning, Statistics, and Earned Value	W. Lipke, M. Jennings	December
Leverage an Estimating Model to Climb the CMM Ladder	A. Minkiewicz	December

PSP/TSP

Building Productive Teams	W. Humphrey	June
Managing Risk with the TSP	D. Webb	June
Making Quality Happen: The Manager's Role	G. Seshagiri	June
PSP: Fair Warning	E. Starrett	June

Risk Management

Both Sides Always Lose: Litigation of Software-Intensive Contracts	T. DeMarco, T. Lister	February
Continuing Risk Management at NASA	L. Rosenberg, T. Hammer, A. Gallo	February
Risk Management Rollout and Installation at the NRO	A. Neitzel, J. Link, R. Barbour, F. Parolek	February
Risk Management: Integrating Crisis into PM Training	K. Knight, D. Corbin R. Hammerly, R. Cox	February
A Practical Approach to Quantifying RiskEvaluation Results	P. Hantos	February
Assessing Software Risk	L. Poulin	December

For a comprehensive search of CrossTalk back issues, please visit the Web site at www.stsc.hill.af.mil/CrossTalk/crostalk.html

BackTalk

Software Pie	G. Petersen	January
Thank Goodness We're Not Dumb Animals	D. Cook	February
Beware the Ides of March Madness	M. Welker	March
State of the Software Industry, Part I	G. Petersen	April
State of the Software, Part II	G. Petersen	May
Common Sense—Can You Dig It?	D. Cook	June
Gilligan's Integration	H. Winward	July
A Tale of Two Monoliths	M. Welker	August
Incredible Suckers	G. Petersen	September
Three Cheers for Big Brother	M. Welker	October
Time to Stuff the Tower, I Mean, Turkey	M. Welker G. Petersen	November
With Great Apologies to Clement Clark Moore	D. Cook L. Dupaix	December

Publisher's Note

Charting Your Course	R. Alder	January
Risky Business	L. Silver	February
Delivering Just-in-Time Training	J. Jarzombek	March
Making an Educated Guess	B. Allgood	April
Study Success, Learn From Failure	R. Alder	May
PSP & TSP–The Necessary Approach	L. Silver	June
Impacting the Future of Process Improvement	B. Allgood	July
What Makes Software Process Improvement?	R. Alder	August
COTS: The Ideal World	L. Silver	September
You Cannot Pass the Buck on Reliable Network Security	B. Allgood	October
Quality Leadership Is the Foundation…	R. Alder	November
Other Disciplines Lend Ideas …	Lt. Col. Palmer	December

With Great Apologies to Clement Clarke Moore …

Twas the night before FY closeout at USAF,
and the PMs were nestled all snug at their desks.

Budgets were adjusted and figured with care
In hopes that funding would soon show up there.

Amounts were looked at, figured with precision.
"Just inflate it 150%" was shouted with derision.

Contractors scurried to minimize their risks
Organic staff tried to amend deliverables lists.

When out on the street there arose such a roar –
I closed down Solitare and ambled over to the door.

I slipped outside, hoping no one would see
(that door was off limits during THREATCON D).

Bright sunshine left me shaking with fear
(with mandatory OT – hadn't seen the sun all year).

Looking down the street, on glistening black tar,
I spied a chubby man climb out of a beat-up old car.

His brow was all furrowed, his face held no cheer.
I knew in a moment – he was a Software Engineer!

His hair was rumpled, his clothes didn't match
his pants drooped, and he showed *just a crack.*

His shirt was un-ironed, and his belly hung low.
His orange tie, blue socks and brown shoes *didn't go.*

Muttering "Sorry I'm late – last meeting over-ran,"
he went inside, and beckoned me to stand.

"I'm from the Pentagon – they sent me here straight.
They're trembling in fear as your schedule is late."

"I'm here to make changes, keep you on your toes.
I'm here to *help* you (you know how it goes …).

"Get moving, and get your budget in the black.
I have some ways to get you back on track.

His lack of knowledge made me hang my head in shame.
He wasn't technical; he just knew technical names.

"Lose Jovial – C++, Ada, or Java is best.
Language is important – does yours pass the test.?"

I tried to argue, to explain our current state.
"Coding isn't our problem – requirements are late."

"We can write software, but our progress is fruitless.
Users won't say what they want – they're all clueless."

He sighed deeply, then shook his head side to side—
"That's been your excuse for years – no more free ride."

"What about our process?" I tried to plea.
But each point I had, he countered with glee.

"We don't mind a process; we like Level 3.
But what we want is for the process to be free.
Who cares about planning, or CMM levels,
as long as developers are coding like devils?"

I couldn't take his attempts to code far too quick.
I explained our process, which made him quite sick.

"We develop it correctly, and have a process to follow.
Requirements with no validation are hollow.

Once we coded blindly, now we see the light –
We follow a process to develop software right!

Design is important (OO does the trick)
because undesigned code cannot be fixed!

Now off you go – back to your boss.
Experience shows – without a process we're lost!"

He muttered curses, but I stuck to my guns,
sending him back to his car on the run.

"You don't understand" he said, driving out of sight.
"You have time to do it over, but not to do it right!"

"I understand well," I said, "I understand fine.
If you want quality software, it just takes time!

I don't want to seem obstinate, rude, nor surly.
But if you want quality, plan for it early."

—Dave Cook, Draper Labs, and Les Dupaix, STSC

Give Us Your Information, Get A Free Subscription

Fill out and send us this form.
OO-ALC / TISE
5851 F Ave., Bldg 849, Rm B-04
Hill AFB, UT 84056-5713
Attn: Heather Winward
Fax: 801-777-5633 DSN: 777-5633
Voice: 801-586-0095 DSN: 586-0095
Or use our online request form at www.stsc.hill.af.mil

NAME:_____

RANK / GRADE:_____

POSITION / TITLE:_____

ORGANIZATION / COMPANY:_____

ADDRESS:_____

_____

BASE / CITY:_____ STATE:_____

ZIP:_____ VOICE: _____

FAX:_____ E-MAIL: _____@_____

CHECK THE BOX(ES) TO REQUEST EXTRA ISSUES FROM 2000:

JAN___ FEB___ MAR___ APR___ MAY___ JUN___

JUL___ AUG___ SEP___ OCT___ NOV___ DEC___

Sponsored by the Computer Resources Support Improvement Program (CRSIP)

CrossTalk / TISE
5851 F Avenue
Building 849, Room B04
Hill AFB, UT 84056-5713
Return Service Requested

PRSRT STD
U.S. POSTAGE PAID
Kansas City, MO
Permit 34