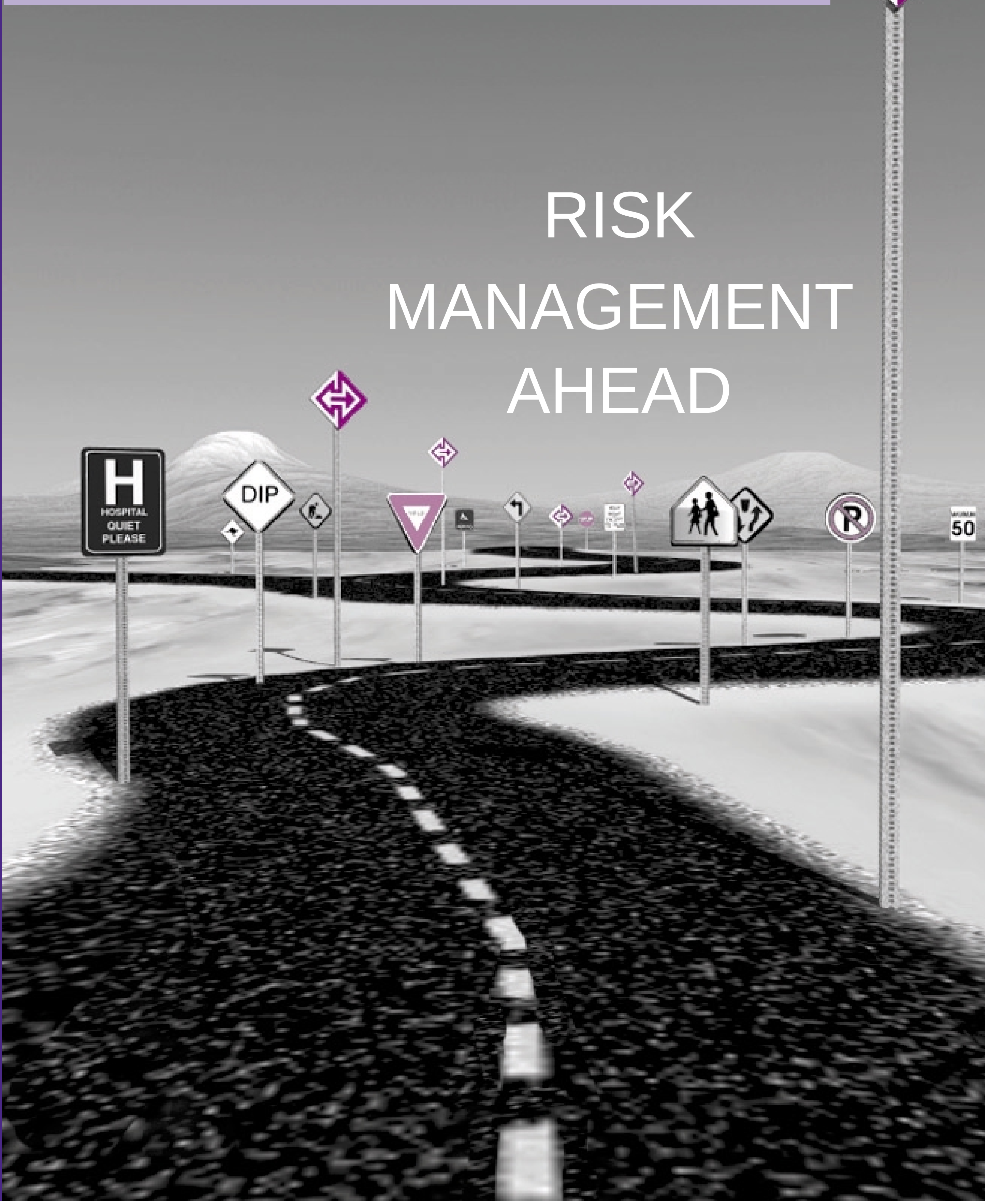
CrossTalk

February 2000 *The Journal of Defense Software Engineering* *Vol. 13 No. 2*

RISK MANAGEMENT AHEAD



On the cover:
Anthony Peters of L-3 Communications in Layton, Utah, represents risk management as a winding road.

Departments

CrossTalk

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

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

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

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

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

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

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

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

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

Risky Business

On a clear, crisp winter Saturday, I experienced risk management at its finest when I took my family to a grand opening of a light rail system that took years of planning and building. Excitement filled the air on this festive day of free trolley rides, and giveaways like hot dogs and T-shirts. There was no parking when we arrived at one of the rail stations, due to the large turnout. I finally left my car in a warehouse parking lot, hoping that no one cared that I had parked in an unauthorized zone (Risk No. 1).

My wife and I and three young children walked to the light rail station and waited. And waited. And waited some more. Three trolleys, stuffed with riders, rolled by before I finally jammed my family onto the next one, wondering as I did so if there would be room on a returning trolley to get us back within the hour (Risk No. 2).

I took my chances and the trolley sped smoothly ahead. I noticed that all the stations along the route were packed with long lines of people waiting to board, and that every trolley we passed was filled to overflowing. When we arrived at the end of the line, the trolley doors rolled open and riders were asked to exit (Risk No. 3).

Many of us looked out the windows at the line of more than 2,000 waiting people and said, "Heck no, we won't go." We knew a long wait was ahead once we exited. There was no way my family would be able to return to our car within an hour (Risk No. 4).

The mutinous group on our trolley cheered as our trolley started back in the other direction. I was happy to think we would soon be back to our waiting car, 15 miles away, with our crying baby. However, the next trolley stop changed the whole afternoon.

We waited for the trolley to move; it did not. After about 10 minutes, passengers were told that the front trolley had brake problems. We waited 30 minutes while they pumped the brakes, a long wait (Risk No. 5) with three children ages 5, 3, and 1.

Eventually, the trolley moved, but at a reduced speed of 2 mph. By the time we arrived at the next station, the trolley's power had shut down due to a software problem (Risk No. 6).

I mitigated my next risk and called a friend, who rescued us. Our risk-filled day ended as we arrived home.

Most of us deal with risks every day. At what point do you execute a risk mitigation plan? In their article, *Continuing Risk Management at NASA,* (see page 7), Dr. Linda H. Rosenberg, Ted Hammer, Al Gallo, and Frank Parolek, point out that risk involves the likelihood that an undesirable event will occur. They write about the importance of risk management in identifying and dealing with potential problems before they reach crisis level, focusing on the project's objective, being proactive, and involving personnel at all levels of the project.

Taking risks is a fact of life. The severity and degree of a risk needs to be properly managed or catastrophic results can occur. Manage risk; do not let it manage you. Be cognizant of the road signs along your project's route that may signal a detour or a bumpy road ahead. Mitigate those risks.

Do not get stranded along the side of the road. No risk management is risky business.

[illegible]

Lynn P. Silver
Associate Publisher

Both Sides Always Lose: Litigation of Software-Intensive Contracts

Litigation of software-intensive endeavors is a major growth industry. The costs of litigation are rising faster than any other aspect of software development. Understanding five key patterns of such litigation is essential to maintaining the health of any organization that builds or buys software.

A contract is a kind of specification. Instead of describing a new system, it describes a business agreement. A contract suffers from some of the same difficulties that plague a specification: neither is ever entirely clear, entirely *right*, or entirely free from interpretation. None of these problems is fatal when there is goodwill and a commonality of interests between the parties. With these two essential ingredients, people work out their differences and come to a successful conclusion. But when the two essential ingredients are missing, that is when matters begin to go south.

Going South to the Tune of More than $100 mil

Take, for example, the case of a contract to build an ambitious reservation system a few years back. The disputants were the Fly-By Information Services Co. and the Magnificent Hotel Corp. (fictitious names used to protect the guilty parties and the authors). Fly-By was to build the system and Magnificent to specify the requirements and pay for the result.

The first sign that all was not right was the extraordinary difficulty of coming up with a contract. As with a specification effort, when contract negotiations are particularly rough, that is a sign that there is conflict brewing beneath the surface. This negotiation was fierce. Everybody hated everybody. It was like one of those marriages that quickly dissolves into bitter acrimony. You have to wonder, when the principals are so wrong for each other, why they bothered. So, too, the contract between Fly-By and Magnificent.

One way to deal with conflict is to paper it over in ambiguity. You do that when you write a speculation that conceals disagreement rather than pleasing one of the disagreeing parties at the expense of another. In an aggressively negotiated contract, the result is not what you would at first expect, a contract with every “i” dotted and “t” crossed. Instead, you tend to end up with ambiguities wherever there are truly unresolvable conflicts. That is what happened in the Fly-By case.

The rest of the story is especially grim. The case was litigated for years and finally settled, with nobody really winning. The settlement was $100 million-plus, but not enough to make anyone whole. No system was built or delivered. Everybody wasted years of their lives. The legal fees were staggering. The opportunity costs, the useful things they could have been doing instead of litigation, were even worse. Both sides showed huge losses.

Since the contract was weak, the case finally turned on a single incident: Fly-By had fired a succession of managers who tried to tell the big boss that the date was unworkable. Magnificent found out, brought in the fired managers as witnesses, then pointed triumphantly to one clearly written contract provision that said Fly-By was obliged to inform its partner if it had credible reason to believe the delivery would be late. That cost Fly-By more than $100 million. Everybody ended up with enormous losses, but Fly-By’s were more than Magnificent’s.

Even told in such sketchy terms, the case of Magnificent vs. Fly-By contained many of the patterns of a typical litigation where software is at the heart. The first of these is obvious to anyone who has ever been involved in software litigation:

Pattern 1: Both sides always lose.

Stepping Back for a Look at the Context

As impressive as growth of the software industry has been, it is outpaced by growth of software-related litigation. It is not unusual for a large software development organization today to have upwards of 50 active cases on its hands. Litigation costs are not usually charged against an information technology budget, but if they were, they would be—when spread across unlitigated and litigated projects—a larger component than coding. We are experiencing an epidemic of litigation on software projects. This is different from the general litigiousness that has often been noted about our society. Americans are all too prone to ask the courts to resolve disagreements, but this has not in the past been particularly true of American corporations. Corporations understood that everybody loses in litigation and that the costs can be ruinous.

The growth of software-related litigation is, we think, due to some factors that affect all corporations today, but have been particularly strongly felt in the software sector. The extensive layoffs in the early 1990s, together with the *lean and mean* attitude they engendered, have led to the glut of litigation today.

The Heart of the Matter

There have been downsizings in our economy before, but the ones that struck us so hard between 1990-95 were curiously vindictive. The people who were booted out were made to feel that they were not just unlucky, but somehow at fault. They were *fat* that needed to be trimmed. The people who remained were also made to feel bad and told in no uncertain terms that they would have to pick up all the work of their dismissed colleagues, and then some, or else risk being booted out.

It often fell to information technology (IT) management to carry out the downsizings and to convey upper management’s attitude of righteous indignation about *fat* on the payroll. The effect on the workers was either catastrophic for those who were laid off, or depressing. That much was obvious. What was not so obvious was the effect on IT management. Our speculation is that the downsizing exercise made many IT managers more fearful and insecure than ever, and caused them to retreat into an attitude of embattled authoritarianism. This attitude has been bad for everyone (except Scott Adams, who has turned it into millions).

What does this have to do with litigation? We believe it is at the very heart of the flurry of litigation that we began to observe by 1995. Embattled IT managers, fearful and under the gun to show improved performance, fell back on lines like. “Do not tell me it cannot be done in two years. I am the boss. It will be done in two years."

Engineers, who knew that deadlines were unworkable or that quality would suffer, shrugged in the face of such insistence. They reasoned that management would learn in the long run that impossible is impossible. And so impossible targets were accepted. They made their way into contracts. The contracts were signed. The project workers tried their best and failed. Then the parties went to court.

This is almost exactly the story of the project that Fly-By conducted for Magnificent. Fly-By wanted the work and bid to win. It obligated itself to perform at a level that time would prove was unattainable. The voice of reason was drowned out by that of management, intent on using its force to impose its wishes. It succeeded, briefly.

Pattern 2: When authority trumps reality, reality always wins in the end.

"Golly, How the Truth Will Out."

An invariant feature of such litigation is that all project records are subpoenaed. That means that each and every manager and worker has to provide the entire contents of his or her files to be copied and provided to the other side. There are no exceptions. You may be reluctant to provide a copy of that bothersome little memo where the true defect rates were discussed, but as the court papers make clear, if you do not provide it and you are found out, you go directly to jail. Not the company, and not just officers of the company, but you. For most employees, this is a fairly persuasive argument. And so, even the most incriminating evidence generally gets delivered.

As litigation consultants and expert witnesses, we spend an inordinate amount of time poring over these project records. The surprising thing is that, in most of these cases, there is a great deal of what we might call *lying.* It is not at all unusual to find yourself with a memo from X to his boss stating that delivery will be delayed by at least six months and a memo from X to the client, dated the same day, providing comforting assurances that the project is on schedule.

Outright lying is a direct sin of commission. Equally common are sins of omission, situations where X knows something bad and neglects to tell it to the client. Omission is a different flavor of lie. Both are subject to this invariant of legal cases:

Pattern 3: Lying to your contract partner is morally indefensible, generally illegal, and always gets found out in litigation.

The old rule of honorable behavior—do not lie to the other guy—seems to have been replaced with a new rule: do not tell the other guy anything that is not true unless he has no possible way of knowing that what you say is not true. Same with sins of omission (what the lawyers call *unfair surprise*), the new rule seems to be that it is OK to neglect to mention an inconvenient fact that the other guy has no way of hearing about on his own. All this subterfuge comes back to eat you alive during litigation.

The Other Side of the Coin

We have spoken mostly of cases where the software builder overcommits. But it is also possible for the other party to be at fault. This happens when a software buyer imposes *wishful thinking* deadlines on a builder, conceals real requirements to keep the price down, and hopes to impose them on the builder as freebie changes, tries to get ambiguity into a contract to be exploited later, and so on. In general, buyers are every bit as likely as builders to try to trump reality with authority, and to tell little lies. And they are just as subject to the three patterns presented.

Buyers are particularly prone to one of the worst fallacies of contracting, the idea that risk always moves with responsibility. It does not. When you are the buyer and another organization agrees to build a system for you, signing the contract moves primary responsibility for successful implementation from you to the builder. Not all the risks involved in attempting the project move with that responsibility (no organization can completely buy its way out of risk). If the contractor fails to deliver, both parties lose. Since this is a real risk from the buyer's point of view, it is incumbent on that buyer to manage that risk. Software is a risky business and both sides of any software project need to do serious risk management. We think that most organizations understand this, but our data indicate that those who get into litigation are disproportionately likely to avoid it:

Pattern 4: Litigation is almost always a result of imperfect risk management by either or both parties.

Most of the litigations we have seen have involved organizations that failed to do any risk managment at all.

Everything we have discussed so far deals with events that precede the actual court case, the causative factors. That is interesting in the abstract, but if you are about to go to court, your mind is understandably occupied with other things, i.e. what do you do to avoid losing your shirt?

You are in a Litigation—What Do You Do?

Litigation may be a lose-lose business, but there are certain defensive strategies that may help to limit your losses. The most important one we know is to investigate and calibrate your project with respect to industry norms. For example, if the other side alleges that you have been fickle about the requirement and heaped ruinous numbers of change requests on the builder, your best defense is to show that your performance in this respect is better than the industry norm. It helps to know that the industry norm for changeability of specification is someplace between 1 and 2 percent per month of the original specified size (measured in function points). It helps to have a good source for such evidence, in this case *The Condensed Guide to Software Acquisition Best Practices* [1].

As you may surmise, a successful use of the norms implies a certain fluency with metrics. Organizations that cannot or do not measure themselves in a fairly systematic way are always at a huge disadvantage in litigation. If you are deficient at measurement, and the other side is on top of it, the jig is up. Metrics is one of the three major subjects on which virtually all litigations turn:

Pattern Five: Most litigations end up focused on measurement, management, or requirements practice, or some combination thereof.

The things you do to win a litigation are doing careful measurements work, focusing on good management practice, and conducting exhaustive and thoughtful analysis of requirements. Paradoxically, these also are three of the principal things you should do to avoid litigation.

A Final Word

The more you think about litigation, the more you must think about the underlying contract. It is tempting to conclude that the lesson here is to get your lawyers involved early and often in order to come up with iron-clad contracts that simply cannot go wrong. While early legal work may be a good investment, it is not entirely realistic to think it can avoid all problems. If the basis of understanding is flawed, no contract will improve it.

Rather than focusing on the contract as a legal instrument, turn your attention to the understanding at the heart of it. When parties develop a real commonality of interest, all is possible. Litigation becomes less likely and success more likely. The project may not go smoothly from beginning to end, but it will tend to respond constructively to the problems it encounters and have the best chance of delivering a useful and effective system.

How can you know whether you are forging a partnership as opposed to signing a deal that will come back to haunt you? The best rule is to think of a contract as bad if one party can win while the other loses. Win-lose situations are a precursor to litigation, since only the most saintly organization could accept lose-lose when it could, under the guarantees of the contract, slip into a win-lose situation. This is the kind of contract you must avoid, even if your organization is the putative winner.

But this sounds crazy. Is not the heart of successful contracting an attempt to place yourself in a position where you can win at the expense of the other party? Too often it is, but such contracts usually turn out badly. The only contract that is truly healthy is one where you feel good signing either side.

That starts you out with the best chance of success. Measure carefully, manage well, and pay attention to requirements.

One more thing: do a little risk management, just in case.◆

Reference

1. *The Program Manager's Guide to Software Best Practices*, Crystal City, Va: Software Program Managers Network, Sept. 1995.

This article has been reprinted with the permission of Cutter Information Corp., provider of information resources for IT professionals worldwide. It originally appeared in *Cutter IT Journal*, April 1998, Vol. XI, No. 4 http://www.cutter.com/consortium

About the Authors

Tom DeMarco and Tim Lister, principals of the Atlantic Systems Guild, have been partners for nearly 20 years. They are the authors of *Peopleware: Productive Projects and Teams*, co-editors of *Software: State of the Art*, and joint designers of seminars on risk management for software; leading successful projects, and controlling software projects. Their consulting practice focuses on project management, measurement, and development methods. They are called on often to serve as expert witnesses and litigation support consultants.

Tom DeMarco
115 Shermans Point Road
P.O. Box 160, Camden, Maine 04843
Voice: 207-236-4735
Fax: 207-236-8432
E-mail: tdemarco@systemsguild.com
Internet: http://www.systemsguild.com

Tim Lister
353 W. 12th St.
New York, N.Y. 10014
Voice: 212-620-4282
Fax: 212-727-1044
E-mail: lister@acm.org

Quote Marks

"Computers are useless, they can only give you answers." —*Pablo Picasso*

"Act in haste and repent at leisure; code too soon and debug forever." —*Raymond Kennington*

"I have traveled the length and breadth of this country and talked with the best people, and I can assure you that data processing is a fad that won't last out the year." —editor in charge of business books, Prentice Hall, 1957

"Software and cathedrals are much the same—first we build them, then we pray." Samuel T. Redwine, Jr.

"I don't think it's that significant." —*Tandy president John Roach, 1981, on IBM's entry into the microcomputer field.*

"I do not fear computers. I fear lack of them." —*Issac Asimov*

"Computers make it easier to do a lot of things, but most of the things they make easier to do don't need to be done."
– Andy Rooney, 60 Minutes

"Computers in the future may weigh no more than 1.5 tons." —*Popular Mechanics,* 1949, forecasting the relentless march of science.

"A computer does not substitute for judgement any more than a pencil substitutes for literacy. But writing without a pencil is no particular advantage." —Robert S. McNamara, *The Essence of Security*

Continuing Risk Management at NASA

The NASA Goddard Space Flight Center (GSFC) Software Assurance Technology Center (SATC) teaches a risk management process based on a course developed in collaboration with the Software Engineering Institute at Carnegie Mellon University. This risk management process has been taught to projects at all NASA Centers and is being successfully implemented on many projects. This paper will discuss the six primary functions of risk management and will give project managers the information they need to understand if risk management is to be effectively implemented on their projects at a cost they can afford.

Software risk management is important because it helps avoid disasters, rework, and overkill, but more importantly because it stimulates win-win situations. Software risk management objectives are to identify, address, and eliminate software risk items before they become threats to success or major sources of rework. In general, good project managers are also good risk managers. It makes good business sense for software development projects to incorporate risk management as part of project management.

NPG 7120.5A, the NASA guidebook for project managers, requires risk management applications and includes a section briefly discussing what should be included in a risk management plan [1]. The SEI-developed course on continuous risk management was first taught in January 1998[1] [2]. Since then, more than 300 students at NASA centers have attended the course.

There are a number of definitions and uses for the term *risk*, but there is no universally accepted definition. What all definitions have in common is agreement that risk has two characteristics:

uncertainty: An event may or may not happen.

loss: An event has unwanted consequences or losses.

Therefore, risk involves the likelihood that an undesirable event will occur, and the consequences can be severe. Risk management can:

- Identify and deal with potential problems before they *are* problems and before a crisis exists.
- Focus on the project's objective and consciously look for things that may affect quality throughout the production process.
- Allow the early identification of potential problems (the proactive approach) and provide input into management decisions regarding resource allocation.
- Involve personnel at all levels of the project; focus their attention on a shared product vision, and provide a mechanism for achieving it.
- Increase the chances of project success.

NASA focuses on continuous risk management that can be applied to any development process: hardware, software, systems, etc. It provides a disciplined environment for proactive decision making to:

- Continually assess what could go wrong.
- Determine which risks are important.
- Implement strategies to deal with them.
- Assure measured effectiveness of implemented strategies.

Risk management must not be allowed to become shelfware. The process must be a part of regularly scheduled, periodic product management. It requires routinely identifying and managing risks throughout all phases of the project's life. The set of continuous risk management functions throughout a project's life cycle is the foundation for the application of continuous risk management. Each risk nominally goes through these functions sequentially, but the activity occurs continuously, concurrently, and iteratively. Risks are usually tracked in parallel while new risks are identified and analyzed, and the mitigation plan for one risk may yield another risk.

Continuous Risk Management Principle Functions

Identify

The purpose of identification is to consider risks before they become problems and to incorporate this information into the project management process. Anyone in a project can identify risks to the project. Each individual has particular knowledge about various parts of a project. Uncertainties and issues about the project are transformed into distinct (tangible) risks that can be described and measured.

During this function, all risks are written with the same, two-part format. The first part is the risk statement, written as a single statement concisely specifying the cause of the concern as well as its impact. The second part may contain additional supporting details in the form of a context.

A risk statement's aim is to be clear, concise, and sufficiently informative that the risk is easily understood. Risk statements in standard format must contain two parts: the condition and the consequence. The condition/consequence format provides a complete picture of the risk, which is critical during mitigation planning. It is read as follows:

given the ***<condition>*** *there is a possibility that* ***<consequence>*** *will occur*

The *condition* component focuses on what is currently causing concern; it must be something that is true or widely perceived to be true. This component provides useful information when determining how to mitigate a risk. The *consequence* component focuses on the risk's intermediate and long-term impact. Understanding the depth and breadth of the impact is useful in determining how much time, resources, and effort should be allocated to the mitigation effort. A well-formed risk statement usually has only one condition, but may have more than one consequence.

Risk statements should avoid abbreviations or acronyms that are not readily understood, sweeping generalizations, and irrelevant detail

Since the risk statement is to be concise, a context is added to provide enough additional information about the risk to ensure that the original intent of the risk can be understood by other personnel, particularly after time has passed. An effective context captures the what, when, where, how, and why of the risk by describing the circumstances, contributing factors, and related issues (background and additional information not in the risk statement).

A diagram of the complete risk statement and context are shown in Figure 1.

An example is shown in Figure 2. One condition and two consequences are the risk statement. The context explains

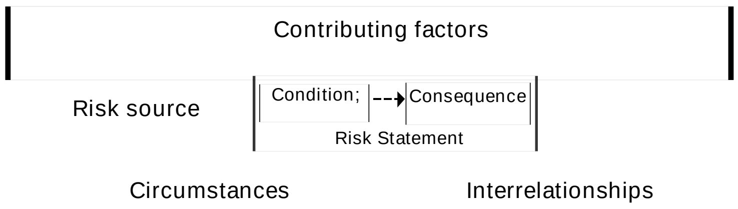


Figure 1. *Risk Statement and Context*

Risk statement: This is the first time that the software staff will use OOD; the staff may have a lower-than-expected productivity rate and schedules may slip because of the associated learning curve.

Context: Object-oriented development is a very different approach that requires special training. There will be a learning curve until the staff is up to speed. The time and resources must be built in for this or the schedule and budget will overrun.

Figure 2. *Example of Risk Statement and Context*

why this is a risk, and supplies additional information for someone unfamiliar with this risk.

Risk identification depends heavily on both open communication and a forward-looking view to encourage all personnel to bring forward new risks and plan beyond their immediate problems. Although individual contributions play a role in risk management, teamwork improves the chances of identifying new risks. It allows personnel to combine their knowledge and understanding of the project.

Analyze

The purpose of analysis is to convert data into decision-making information. Analysis is a process of examining the risks in detail to determine the extent of the risks, how they interrelate, and which ones are the most important. Analyzing risks has three basic activities: evaluating their attributes (impact, probability, and time frame), classifying, and prioritizing or ranking them.

Evaluating—The first step provides better understanding of the risk by qualifying the expected impact, probability, and time frame. This involves establishing values for:

Impact: the loss or negative affect on the project should the risk occur

Probability: the likelihood the risk will occur

Time frame: the period when action must be taken in order to mitigate the risk

Figure 3 shows sample values used to evaluate a risk's attributes.

Classifying—The next step is to classify risks. There are several ways to classify or group risks. The ultimate purpose of classification is to understand the nature of the risks facing the project and to group any related risks to build more cost-effective mitigation plans. The process of clarifying risks may reveal that two or more risks are equivalent—the statements of risk and context indicate that the subject of these risks is the same. Equivalent risks are duplicate statements of the same risk and should be combined into one.

Prioritize—The final step in the analysis function is to prioritize the risks. The purpose is to sort through a large number of risks and determine which are more important—the few vital risks—and to separate the risks to be dealt with first when allocating resources. This involves partitioning risks or groups of risks based on the *vital few* sense and ranking risks or sets of risks based on consistently applying an established set of criteria. No project has unlimited resources with which to mitigate risks. It is essential to determine consistently and efficiently which are more important and then to focus those limited resources on mitigating risks.

Conditions and priorities will change during a project, and this natural evolution can affect the important risks to a project. *Risk analysis must be a continual process.* Analysis requires open communication so that prioritization and evaluation are accomplished using all known information. A forward-looking view enables personnel to consider long-range impact.

Plan

Planning is the function of deciding what, if anything, should be done about a risk or set of related risks. Decisions and mitigation strategies are developed based on current knowledge of project risks. The purpose of plan is to:

- Make sure the consequences and sources of risk are known.
- Develop effective plans.
- Plan efficiently (only as much as needed or is of benefit).
- Produce the correct set of actions over time that minimize the cost and schedule impacts of risks while maximizing opportunity and value.
- Plan important risks first

There are four options to consider when planning for risks:

1. Research: establish a plan to research the risk(s).
2. Accept: decide to *accept* the risk(s), and document the rationale behind the decision.
3. Watch: monitor risk conditions for any indications of change in probability or impact (tracking metrics must be established and documented).
4. Mitigate: allocate resources and assign actions in order to reduce the probability or potential impact of risks. This can range from simple tasking to sweeping activities:
   - Action Items: a series of discrete tasks to mitigate risk.
   - Task Plan: formal, well-documented and larger in scope.

Attribute	Value	Description
Probability	Very Likely (H) Probable (M) Improbable (L)	High chance of this risk occurring, thus becoming a problem > 70% Risk like this may turn into a problem once in a while {30% < x < 70%} Not much chance this will become a problem {0% < x < 30%}
Impact	Catastrophic (H)	Loss of system; unrecoverable failure of system operations; major damage to system; schedule slip causing launch date to be missed; cost overrun greater than 50% of budget
	Critical (M)	Minor system damage to system with recoverable operational capacity; cost overrun exceeding 10% (but less than 50% of planned cost)
	Marginal (L)	Minor system damage to project; recoverable loss of operational capacity; internal schedule slip that does not impact launch date cost overrun less than 10% of planned cost
Timeframe	Near-term (N) Mid-term (M) Far-term (F)	Within 30 days 1 to 4 months from now more than 4 months from now *NOTE: refers to when action must be taken*

Figure 3. *Sample Attribute Values*

Dealing with risk is a continuous process of determining what to do with new concerns as they are identified and efficiently utilizing project resources. An integrated approach to management is needed to ensure mitigation actions do not conflict with project or team plans and goals. A shared product vision and global perspective are needed to create mitigation actions on the macro-level to benefit the project, customer, and organization. The focus of risk planning is to be forward-looking, to prevent risks from becoming problems. Teamwork and open communication enhance the planning process by increasing the amount of knowledge and expertise applied to the development of mitigating actions.

Track

Tracking is the process by which risk status data are acquired, compiled, and reported. The purpose is to collect accurate, timely, and relevant risk information, and to present it in a clear and easily understood manner to the appropriate group of people. Tracking is done by those responsible for monitoring *watched* or *mitigated* risks. Tracking status information becomes critical to performing the next function in the continuous risk management paradigm, i.e. control. Supporting information, such as schedule and budget variances, critical path changes, and project/performance indicators can be used as triggers, thresholds, and risk- or plan-specific measures where appropriate.

When a mitigation plan has been developed for a risk or risk set, both the mitigation plan and the risk attributes are tracked. Tracking the mitigation plan, or even a list of action items, will indicate whether the plan is being executed correctly and/or on schedule. Tracking any changes in the risk attributes will indicate whether the mitigation plan is reducing the impact or probability of the risk. Tracking risk attributes gives an indication of the effectiveness of the mitigation plan.

Program and risk metrics provide decision makers with information needed for making effective decisions. Normally, program metrics are used to assess the cost and schedule of a program as well as the performance and quality of a product. Risk metrics are used to measure a risk's attributes and assess the progress of a mitigation plan. They can also be used to help identify new risks. For example, a program metric might look at the rate of module completion. If this indicates that the rate of completion is lower than expected, then a schedule risk should be identified.

Open communication regarding risk and mitigation status stimulates the project and risk management process. Tracking is a continuous process—current information about a risk status should be conveyed regularly to the rest of the project. Risk metrics provide decision makers with information needed for effective decisions.

Control

The purpose of the control function is to make informed, timely, and effective decisions regarding risks and their mitigation plans. It is the process that takes in tracking status information and decides what to do based on the reported data. Controlling risks involves analyzing status reports, deciding how to proceed, and implementing the decisions.

Decision makers need to know when or whether there is a significant change in risk attributes, and the effectiveness of mitigation plans within the context of project needs and constraints. The goal is to obtain clear understanding of the current status of each risk and mitigation plan relative to the project in order to make decisions based on that understanding. Tracking data is used to ensure that project risks continue to be managed effectively and to determine how to proceed with project risks. Options include:
Replan: A new or modified plan is required when the threshold value has been exceeded, analysis of the indicators shows that the action plan is not working, or an unexpected adverse trend is discovered.
Close the risk: A closed risk is one that no longer exists or is no longer cost-effective to track as a risk. This occurs when the probability or impact falls below a defined threshold, or the risk has become a problem and is tracked.
Invoke a contingency plan: A contingency plan is invoked when a trigger has been exceeded or other related action needs to be taken.
Continue tracking and executing the current plan: No additional action is taken when analysis of the tracking data indicates that all is going as expected or project personnel decide to continue tracking the risk or mitigation plan as before.

Open communication is important for effective feedback and decision making, a critical aspect of control. Risk control is also enhanced through integrated management; combining it with routine project management activities enables comprehensive project decision making.

Communication, Documentation

The purpose of communicating and documenting is for all personnel to understand the project's risks and mitigation alternatives as well as risk data to make effective choices within the constraints of the project. Communication and documentation are essential to the success of all other functions within the paradigm and are critical for managing risks.
Identify: In risk identification, risk statements are communicated.
Analyze: In analysis, project personnel communicate information about impact, probability, and time frame attributes. Risk classification involves grouping risk information communicated by individuals.
Plan: Action plans are developed and communicated to project personnel.
Track: Reports designed to communicate data to decision-makers are compiled.
Control: The decisions made during control must be communicated and recorded to project personnel.

For effective risk management, an organization must have open communication and formal documentation. Communicating risk information is often difficult because the concept of risk comprises two subjects that people do not normally deal well with: probability and negative consequences.

Not only continuous risk management, but the project as a whole is in jeopardy when the environment is not based on open communication. No one has better insight into risks than project personnel, and management needs their input. Experienced managers know that the free flow of information can make or break any project. Open communication requires:

- Encouraging free-flowing information

at and between all project levels.

- Enabling formal, informal, and impromptu communication.
- Using consensus-based processes that value the individual voice, bringing unique knowledge and insight to identifying and managing risks.

NASA Risk Management Course

Risk is a daily reality on all projects, and continuous risk management should become just as routine. It should be ongoing and comfortable, and neither imposed nor forgotten. Like any good habit, it should seamlessly fit into the daily work. During the course taught at NASA, various tools and methods are demonstrated that will work for any project. The key is to adhere to the principles, perform the functions, and adapt the practice to fit the project's needs. Continuous risk management is not one-size-fits-all. To be effective, tailoring is needed. Tailoring occurs when organizations adapt the processes, and select methods and tools which best fit their project management practice and their organizational culture. Following the principles of continuous risk management is the key to successful tailoring.

With this in mind, the Continuous Risk Management course for NASA was tailored to two days. The first day was lecture, covering all material with some exercises applying methods and tools. This is an intense day, as there is a lot of information to absorb. The second day is devoted to a project workshop. In most classes, personnel from one or two projects attend the lecture, then split up for the workshop (Classes are limited to 20 students.) The workshop is done in small groups. Periodically, these groups come together to review what each group has chosen to work on. Depending on the audience, there are two possible workshops, one for management, and the other for the implementation team.

The workshop for management starts by compiling the project information needed for the risk management plan. This starts with getting the functional organizational chart, identifying key meetings where risk management activities should take place, and identifying key personnel. The methods and tools to be used are then selected, and the criteria for the attributes probability, impact, and time frame are defined. This usually takes two to three hours. A shortened version of the implementation workshop described below is then applied.

The implementation workshop starts by identifying risks to the project based on everyone's knowledge. Phrases are used, with brainstorming, to compile a list of more than 20 potential risks. It is stressed that if it is a problem now, it is not a risk. From this list five risks are identified as those the group feels it can do something about and would like to work on. The risks are written using the correct format of condition and consequence as shown in Figure 1. The risk context is discussed but not written. Using these five risks and the attribute definitions from management, the risks are classified and prioritized. A mitigation plan for the top risk is developed, data for tracking is identified, and presentation formats discussed. Depending on time, two or three risks are processed through this cycle so that the attendees not only feel comfortable with the process, they have some risks specific to their project that they can start working on. Based on course feedback, it seems the workshop is the key to the training's success.

When a class is not made up of people from the same project, either the group is told to make up a project based on common experience, or it uses a project with which many are familiar. The second option is encouraged so real work is accomplished, although it only benefits a few attendees.

After completion of the course, students should:

- Understand the concepts and principles of continuous risk management and how to apply them.
- Possess basic risk management skills for each function of the risk management paradigm.
- Be able to use key methods and tools.
- Be able to tailor CRM to a project or organization.

Implementation

Three steps should be considered when implementing risk management. First, project risk management should be structured. The training itself is not important, it is what the training does for the project. The training helps the project to see how a formal process can be used to manage risks, but more importantly facilitate communication and initial brainstorming among project personnel.

Second, the project should adopt tools that project members are familiar with to aid in tracking risks and communication of risk status. The key is to use tools that members know how to use, and that they will use.

Lastly, the risk management process needs to be integrated into the normal project management process. Risk management must become the normal way of doing project business. This ensures that, rather than a separate process requiring extra overhead, risk management is ingrained. This leads to a cost-effective implementation within the project.

Conclusion

Most project managers agree that risk management works, but the difficulty lies in actual implementation, even when it is required. The risk management plan is often hastily written and then thrown in a corner to gather dust. In addition to the course, NASA has established a web site, http://satc.gsfc.nasa.gov/crm, that contains sample risk management plans and a schedule of classes. Much time is spent discussing with managers the benefits of taking a formal training course, the cost of which is more than recovered by a project when team members all work toward common goals in a coordinated manner.◆

References

1. *Continuous Risk Management Guidebook,* CMU, SEI, 1996
2. *NASA Procedures and Guidelines* 7120.5A, section 4.2.

Note

1. Some material is based on reference No. 1.

About the Authors

Linda H. Rosenberg manages the Software Assurance Technology Center at Goddard Space Flight Center, NASA. The SATC primary responsibilities are in the areas of metrics, assurance tools and techniques, risk management, and outreach programs. Although she oversees all work areas, Rosenberg's area of expertise is metrics. The emphasis of her work with project managers is metrics application to evaluate quality of development products. She holds a bachelor's degree in mathematics and a master's and a doctorate degree in computer science.

Al Gallo manages the Software Assurance Technology Center at NASA's Goddard Space Flight Center. He has more than 15 years experience in software systems engineering and quality assurance. As one of the SATC's lead trainers of continuous risk management, he has provided training and consulting throughout NASA. Gallo holds bachelor's degrees in pure mathematics and computer science as well as a master's degree in technical management from the Johns Hopkins University, Baltimore, Md.

Ted Hammer is the Associate Chief of the Systems Safety and Reliability Office at the NASA Goddard Space Flight Center. His duties entail overall management responsibility for the continuous risk management and SATC activities of the office. The SATC is the GSFC center of excellence for applied research in software assurance tools and methods. Hammer has more than 22 years experience in software development and assurance, and holds a bachelor's degree in electrical engineering from the University of Maryland. He is a member of the American Society for Quality.

Frank Parolek is the SATC Senior Risk Management Coordinator at NASA's Goddard Space Flight Center. He is responsible for providing continuous risk management (CRM) training and other risk management consulting at all NASA sites. He also coordinates the CRM Train the Trainers program and has provided training to the FAA, Army, and other organizations external to NASA. Parolek earned a bachelor's degree in liberal arts from Regents College and an advanced Russian Linguist Certificate from Defense Language Institute/ Foreign Language Center.

Contact the authors of this article at http://satc.gsfc.nasa.gov/personnel/index.html

Call for Articles

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

Personal Software Process and Team Software Process
July 2000
Submission deadline: March 1

Object-Oriented Technology
August 2000
Submission deadline: April 3

Software Acquisition
September 2000
Submission deadline: May 4

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

Please follow the *Guidelines for CROSSTALK Authors*, available on the Internet at http://www.stsc.hill.af.mil. Send submissions to:

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

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

Letters to the Editor

I would like to compliment Norm Brown on his paper published October 1999, *High-Leverage Best Practices—What Hot Companies are Doing to Stay Ahead and How DoD Programs Can Benefit.* Could you follow it up with a paper describing and listing the 16 critical software practices?

I think this would be a big help to those industry and government people who develop and maintain software.

Dr. L.G. Egan
Software Certification Institute

Editor's Note

Please see our October issue on best practices, in which Jane T. Lochner of the Navy addresses the "16 Critical Software Practices for Performance-Based Management." Thank you for writing.

May I suggest another entry to *Influential Men and Women of Software?* (CROSSTALK, December 1999)

The women who programmed ENIAC were given the task of programming the first modern electronic computer. Most of what we now consider good programming was invented by these "amateurs," including notions of subroutines and software development process.

We who make our living programming computers owe a debt to the creativity of these unknown women, a tribute to whom may be found in a recent exhibit at the Smithsonian (American History) in Washington, D.C.

Joe Iaquinto

Talk to CROSSTALK

We welcome reader comments regarding CROSSTALK articles or matters pertaining to software engineering. Please send your comments and Letters to the Editor to crosstalk.staff@hill.af.mil or mail to

OO-ALC/TISE
Attn: CROSSTALK staff
7278 Fourth St.
Hill AFB, Utah 84056-5205

Please limit letters to less than 250 words. Include your name, phone number, and

Integrating "Crisis" into Project Management Training

Risk management is, and should be, a hot topic in software engineering, given the frequency and severity of the failed and badly delivered software projects. Though more literature is being published on the topic, the frequency of software delivery problems does not appear to be dropping. In this article we review a few of the best software risk management literature and software tools.

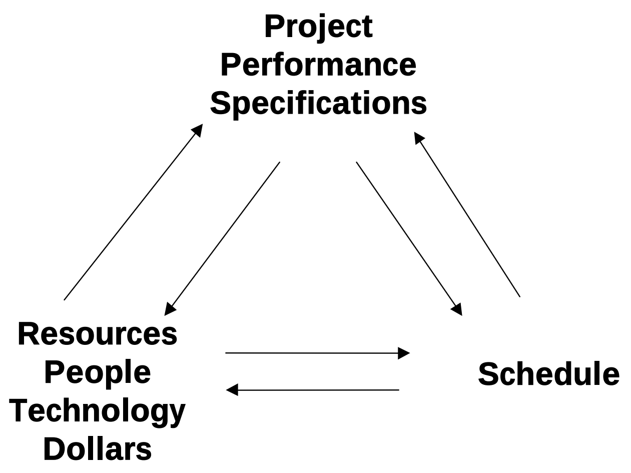
Our conclusion is that the problem does not appear to be in the body of knowledge, but in project management training. Our solution is to introduce "crisis" into the training process. The result is that risk is always present; bad events occur. Risk management is part of every project manager's daily repertoire of project management skills. The key is to train project managers on the determination, assessment, and quick resolution of the unexpected event that creates the disruption.

The Age Old Problem—Software Projects Fail

How many times have you been part of this scenario? Your boss informs you that, with no apparent warning, the software project will be months late, cost more than twice the budget, not meet the required performance, perform an unnecessary task, or be scrapped because it cannot be finished.

The fact that there are software problems is a given. The realization of how poorly we deal with them is of great concern to all of us in the software engineering business. Software problems result because there are always risks involved in the design, creation, and implementation of software. Risks result from incomplete understanding of the project during the initial project planning. Other risks arise from events that occur as projects progress.

Let us consider a simplified scheme of each project. The project management process involves balancing of the three key project components: performance specifications; resources, which include people, technology and dollars, and schedule. Each project manager has the balancing act of assuring that performance can be achieved with resources at the project's disposal and within the time allowed for completion. The project planning process is to carefully review each component to assure compatibility. During this planning process, any change in one of the three key project components requires re-evaluation and realignment of the other two.


Project Performance Specifications
Resources People Technology Dollars
Schedule

The Project—The Balancing Act

Project managers are usually trained with the implicit assumption that the world is fully understood and predictable. They are taught to develop their plans as if project specifications are fixed and well-defined; resources needed are understood, and once-specified, constant; and the agreed-upon schedule will not change. This is a perfect world.

Unfortunately, we in software engineering do not see this perfect world in the projects we encounter. We live in a world where frequent change is required in the scope of a project. Resources we had counted on do not materialize or are inappropriate. The initial schedule is accelerated as the project becomes critical to the firm's success. We must analyze and plan for risk. Substantial risk management literature has developed that discusses ways to manage risk.

Literature on Software Engineering Risk Management

As software engineering practices have matured, there has been great improvement in available tools. The Software Engineering Institute (SEI) at Carnegie-Mellon University has developed the Software Capability Maturity Model® to use as a guide for good practices. The Project Management Institute has published the Body of Knowledge for Project Management that can be utilized in software engineering. While there are several methodologies that can be adapted to any project, the question remains, "Are these methodologies sufficient to offset the risk found in software engineering?" Growing literature on risk management tells us no.

Three publications stand out as central to the body of knowledge for software engineering risk management:

1. SEI's *Continual Risk Management Guidebook* [1].
2. *Tutorial: Software Risk Management* by Barry W. Boehm [2].
3. *Software Engineering Risk Management* by Dale W. Karolak [3].

These authors would generally agree on the following definitions:

Risk—The possibility that a development project incurs a loss as a result of inadequate quality of final system, higher costs, schedule delay, or total failure.

Risk Management—The discipline to discover and eliminate software engineering risk, so that it ceases to threaten the success of a software project.

Boehm's book contains a chapter by Tom Gilb, "Principles of Software Engineering Management" with some provocative quotes on why risk management is important:

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

"The risk principle: If you do not attack the risks, they will actively attack you."
"The risk prevention principle: Risk prevention is more cost-effective than risk detection."
"The risk sharing principle: The real professional is one who knows the risks, their degree, and their causes, and the action necessary to counter them, and shares this knowledge with his colleagues and clients."
"The asking principle: If you do not ask for risk information, you are asking for trouble."
"The principle of risk exposure: The degree of risk, and its causes must never be hidden from decision-makers."

SEI's *Continual Risk Management Guidebook* is a helpful volume that provides an extensive and disciplined proactive decision framework to:

1. Assess continuously what could go wrong (risks).
2. Determine which risks are important to deal with.
3. Implement strategies to deal with those risks.

Tutorial: Software Risk Management develops Boehm's framework for software engineering risk management by presenting a set of methods around risk assessment and control.

- Risk Assessment
  - Risk Identification
  - Risk Analysis
  - Risk Prioritization
- Risk Control
  - Risk Management Planning
  - Risk Resolution
  - Risk Monitoring

Software Engineering Risk Management by Karolak uses an interview technique to develop software risk metrics. He presents six risk management activities:

- Risk Identification
- Risk Strategy and Planning
- Risk Assessment
- Risk Mitigation/Avoidance
- Risk Reporting
- Risk Prediction

The SEI, Boehm and Karolak models are quite similar in their general approach. They treat software engineering risk as a planned process. They all have a risk management process that contains a set of repeated steps. These steps focus on risk identification and risk analysis with a clear risk-reporting process, which serves as a means to communicate risks to individuals and organizations involved. Both the SEI and Karolak frameworks contain a risk taxonomy and risk metrics.

We live in a world where frequent change is required in the scope of a project.

While these three frameworks are extremely helpful, they present a more predictable world than the one we encounter in software engineering professional careers. It is almost always the unexpected that creates the crisis and generates the project risk.

We have found that it is impossible to provide a list or description of the risks that we might encounter. And when we have, what turns out to be the killer risk was not on the list: a key employee leaves unexpectedly, the client changes the scope without warning, the technology we thought would work does not, a corporate crisis results in a dramatic reduction in the project's resources, the project schedule is shortened because of changing corporate strategy.

The risk first looms as a crisis that the project team is ill-prepared to handle. The SEI, Boehm, or Karolak frameworks are excellent, but we argue that they are not enough for two reasons. First, many risks will not be definable until a crisis hits. Second, we need to train project managers to be reactive problem solvers as well as thorough risk planners.

Risk Training—Project Planning Simulated Crisis

Introducing crisis into the formal project management training works. Let us move away from the view of project management dealing with a known and predictable world as we introduce individuals to principles of project management. Let us face up to the complexity of the world of software engineering, whether dealing with the development of a new system or the upgrade of an existing one. Risks that create the project crisis are never fully predictable.

Moving to a project management training that introduces the unexpected in the training project or simulation prepares the future project managers for real world reality. Project managers in training are forced to confront the three key project components—performance specifications, resources, and schedule—and rebalance them given the crisis situation. You deal with the actual or potential crisis recognizing that it is a normal and an expected aspect of the project management process. The project management training sessions provide actual training experience on the second aspect of the SEI, Boehm, or Karolak frameworks. You incorporate a crisis in the learning situation that forces the participants to solve an actual problem (risk) that has occurred.

Conclusion

Software engineering contains and will be fraught with risk through the foreseeable future. Risk management provides a useful preplanning perspective. Unfortunately, with the creation of a software engineering project plan, risk frequently cannot be specified during early stages of a project. Learning how to quickly and appropriately respond to an unexpected major alteration in performance specifications, resources, or schedule needs to be instilled into the project manager through training. Training project managers with random crisis situations in course projects works. The training curriculum may be more difficult to create, but the value in developing project managers who can successfully cope and understand risk is worth the effort.◆

References

1. Dorofree, Audrey J.; Walker, Julie A.; Alberts, Christopher J.; Higuera, Ronald P.; Murphy, Richard L.; and Williams, Ray C.; *Continuous Risk Management Guidebook.* Carnegie-Mellon University, 1996.
2. Boehm, Barry W. *Tutorial: Software Risk Management.* IEEE Computer Society Press, 1986.
3. Karolak, Dale W. *Software Engineering Risk Management.* IEEE Computer Society Press, 1996.

About the Authors

Dr. Kenneth Knight is professor of information systems management at Seattle Pacific University. Knight has been active as an author, professor, manager, and consultant in the information systems area for 35 years. His previous academic positions were in the School of Business at The University of Texas at Austin and Stanford University.

Seattle Pacific University
3307 3rd Ave. West
Seattle, Wash. 98119
Voice: 206-281-2906
Fax: 206-281-2733
E-mail: kknight@sup.edu

Darrell Corbin is an associate technical fellow in Companywide Software Engineering at the Boeing Co. with more than 31 years experience in business and engineering information systems. He provides Boeing-wide support in software methodologies and software process improvement. He also conducts reviews and evaluations of IS suppliers, projects, and organizations. He has bachelor's and master's degrees in economics from Washington State University, and is a Certified Computing Professional.

Russ Hamerly is project manager in Distributed Computing Program Integration at the Boeing Co. He has more than 25 years experience in electronic switching, networking, and computing systems. He provides Boeing-wide support in program management, system management, and emerging workstation technologies. He is a subject matter expert for the SRP web site. He has a bachelor's degree from the University of Washington and a master's degree in business administration from the University of Notre Dame.

Roger Cox is a senior manager in engineering information systems at Boeing Commercial Airplanes. He has more than 32 years of information technology experience, including the last 20 years at Boeing. Cox also serves as an adjunct professor of software engineering at the University of Washington and an adjunct professor of information systems in the graduate program in the school of business at Seattle Pacific University. Cox holds bachelor's degrees in engineering physics, mathematics, and computer science, and master's degrees in physics and computer science, is doing doctoral studies in computer science, and holds an advanced management certificate. Cox is a retired Lt. Col. in the Air Force, where he served in various manugacturing, engineering, and information technology organizations.

The Boeing Co.
P. O. Box 3707
Seattle, Wash. 98124-2207
Voice: 206-655-2121
Corbin's e-mail: darrell.s.corbin@boeing.com
Hamerly's e-mail: russell.p.hamerly@boeing.com
Cox's e-mail: roger.l.cox@boeing.com

Coming Events

March 6-8
13th Conference on Software Engineering Education and Training CSEET&T
http://www.se.cs.ttu.edu/CSEET2000

March 6-10
Software Management/Applications of Software Measurement SM/ASM 2000
http://www.sqe.com/smasm/2000/

March 13-17
6th International Conference on Practical Software Techniques
http://www.softdim.com

March 20-22
5th Annual Association for Configuration & Data Management Technical Conference
http://www.acdm.org

March 20-23
12th Software Engineering Process Group Conference
SEPG 2K
http://www.sei.cmu.edu/products/events/sepg/

April 11-14
Infosecurity 2000
http://www.infosec.co.uk/page.cfm

April 18-20
FOSE: Leading-Edge Technology for Leaders in Government
http://www.fedimaging.com/conferences

April 24-28
SEA 2000, held in Canberra, Australian Capital Territory
E-mail for information: johnl@sea-act.com.au

See page 20 for STC 2000, April 30-May 4

June 20-22
Product Focused Software Process Improvement PROFES 2000
http://www.ele.vtt.fi/profes2000

August 28-31
1st Software Product Line Conference SPLC1
http://www.sei.cmu.edu/plp/conf/SPLC.html

w i crossed e s

Alan Turing, who in the 1930s developed what became known as the Turing Machine, died in 1954. *Influential Men and Women of Software,* in the December issue was, regrettably, in error.

A Practical Approach to Quantifying Risk Evaluation Results

There is a vast literature documenting approaches and tools that address risk assessment and mitigation. In this paper, "hard" and "soft" classifications are introduced, that are based on either the mathematical rigor describing the development of the model or the mathematical rigor expected from the user during the use of the tool. The goal is to present a simple, practical approach to risk analysis, combining the identified benefits, without suffering from the known liabilities. The solution presented here is a combination of the Software Engineering Institute/Software Risk Evaluation (SEI/SRE) method, and Constructive Cost Model (COCOMO).

Software risk management, if practiced properly, is a set of continuous activities for identifying, analyzing, planning, tracking and controlling risks, which is conducted in the context of daily project management. A project planner's first reaction may be to avoid risks all together, but relying strictly on avoidance as a risk mitigation technique is usually inadequate.

Project success primarily depends on the ability to manage the delicate balance of opportunities and risks. Unfortunately, when all risk goes away, so does opportunity. Since risks ultimately manifest themselves incrementally as unexpected cost elements, risk management can also be viewed as a way to dynamically handle the cost/benefit analysis of a project. While techniques for risk identification are usually handled separately from software cost estimation, cost aspects of risks can be used as a communication vehicle during risk prioritization. It has also been determined that parametric cost estimation models are well-suited for risk evaluation [1].

The term *parametric* refers to the fact that the cost is determined via the use of algorithms operating on the parameters of mathematical equations. This structure makes parametric models the prime candidates for carrying out rapid, what-if sensitivity analysis of the cost drivers. Due to their inherent characteristics, nonparametric or nonalgorithmic models, such as expert judgment or estimation by analogy, are not well fitted for sensitivity analysis. This leads to our main proposal, i.e., making the connection between an established risk assessment tool (SRE) and an industry-wide accepted parametric software cost model and estimation tool (COCOMO II).

Risk Management

Based on Barry Boehm's work [2], the risk management steps are outlined in Figure 1.

RISK ASSESSMENT	RISK CONTROL
Risk Identification	**Risk Management Planning**
Checklists	Buying Information
Decomposition	Risk Avoidance
Decision Driver Analysis	Risk Transfer
Assumption Analysis	Risk Reduction
Risk Analysis	Risk Element Planning
Performance Models	Risk Plan Integration
Cost Models	**Risk Resolution**
Network Analysis	Prototypes, Simulations
Decision Analysis	Benchmarks
Quality Factor Analysis	Staffing
Risk Prioritization	Analysis
Risk Exposure	**Risk Monitoring**
Risk Leverage	Milestone Tracking
Compound Risk Reduction	Top-10 Tracking
	Risk Reassessment
	Corrective Action

Figure 1. *Risk Management Steps*

This paper focuses on the connection between software risk identification and cost-model based risk analysis, using risk exposure as a prioritization tool. (The taxonomy based questionnaire, which will be discussed in detail later, is basically a checklist.). Please note that this approach permits the determination of cost ramifications of risks only in the software development domain. Other very quantifiable business risks, such as loss of market opportunity, and loss of sales, can be determined from software development data, but cannot be automatically computed. Similarly, tools can provide quantification of risks, but the overall prioritization and resolution has to be done in the full context of project management.

Risk Taxonomies

Generally defined, software risk taxonomy[1] provides a basis for systematically organizing and analyzing risks in a software project. *Risk Taxonomies*, is intentionally plural, because in addition to describing the importance of a specialized risk taxonomy, we also want to note a level of what we consider undesired proliferation of software risk taxonomies.

Overview of Risk Taxonomy Related Articles

Without assuming completeness, a brief description of current articles follows, where overt or covert development[2] of risk taxonomies plays a role:

- The SEI report lays the foundation of the development of the SEI taxonomy, and discusses the basic concepts of risk taxonomies [3].
- In P.R. Garvey's presentation, the risk elements are described in risk templates, and the taxonomy is implemented via web-based links [4].
- T. Moynihan chose to elicit constructs from experienced managers to determine how they assess risk, after deciding that the taxonomies published in the literature were inadequate [5].
- H. Barki et al. conducted a wide review of the literature and determined 35 variables that were used as taxonomy for risk assessment [6].
- R.J. Madachy developed an extensive rule-based system (identifying 600 risk conditions), where rules were structured around COCOMO cost factors, reflecting an intensive analysis of the potential internal relationships between cost drivers [7].
- K. Känsälä built his tool around 15 risk items he identified as critical, after filtering the data received from 14 selected companies [8].
- E.H. Conrow and P.S. Shishido documented experiences on large projects at TRW, and defined taxonomy consisting of 17 software risk issues [9].
- At Xerox, the SEI-developed taxonomy and the SRE method [10] was evaluated and used in five major projects. While the taxonomy does not provide a complete coverage for all

SEI Software Risk Evaluation Method

The scope of the SRE method is identification, analysis, and preliminary action planning for mitigation. The software risk taxonomy (See Appendix) provides a consistent framework for risk management. It is organized on the basis of three major software risk classes: product engineering, development environment, and program constraints.

Risk elements of these classes are identified at the next level, which are further decomposed into risk attributes. SEI also developed a taxonomy-based questionnaire to carry out structured interviews by an independent assessment team. A sample, customized segment of the TBQ is shown in Figure 2. (The numbering of the questions refers to the original numbering in the full, complete SEI documentation).

Risks are identified and recorded in interviews. After interviews, based on perceived severity and probability of occurrence, the assessment team determines risk magnitude ratings. (Figure 3.)

Note that the relevance of the shaded rows is explained later, when this risk report sample is used to demonstrate the new process. First the assessment team will filter, consolidate and interpret the results. Every risk item is rated separately by the assessment team members (A, B and C in the example), using the risk magnitude matrix (Figure 6.) Severity and probability are separately rated on a scale from one to three, and risk magnitude is computed as severity times probability. This results in a one to nine numerical rating, where one represents improbable and marginal risks, and nine represents risks considered very likely and catastrophic.

Class: A. Product Engineering
Element: 2. Design & Implementation
Attribute: d. Performance

Starter question: [22] Are there any problems with performance?

Cues:
Throughput
Scheduling asynchronous events
Real-time responses
Impact of hardware/software partitioning

Starter question: [23] Has a performance analysis/simulation been done?

Follow-up questions: (YES) (23.a) What is your level of confidence in the results?
(YES) (23.b) Do you have a model to track performance?

Figure 2. *Taxonomy Based Questionnaire Sample*

situations, combining it with a customized SEI Taxonomy-based Questionnaire (TBQ) makes it the preferred tool for assessing risks in the majority of software projects [11].

We found that all authors decided that the introduction of new risk categories, or the creation of a whole new taxonomy, was needed. In our opinion, this is not always justified. During the pilot of the SRE method at Xerox, the SEI taxonomy was criticized in two areas. In large software projects, respondents complained that TBQ terms and language did not always map to local terminology (for example, the classification of contractor relationships). Second, respondents from firmware development projects stated that in their work the distinction between hardware and software was somewhat blurred, and consequently their risk issues were not always adequately covered.

Conclusion Drawn from the Literature Review

It seems that the application of any risk taxonomy always requires a certain level of customization before use, and the quest for the perfect taxonomy, consequently the perfect risk management tool, is fruitless. Also, in the case of actual, computer-based tools, eventually the taxonomy ends up hard-coded into the tool. Instead of further specialization, the approach should be exactly the opposite; we should step back and find a framework that is applicable for a large class of projects, with the understanding that a certain level of customization will take place. As stated earlier, the SEI taxonomy satisfied this requirement.

Cost Estimation with COCOMO II

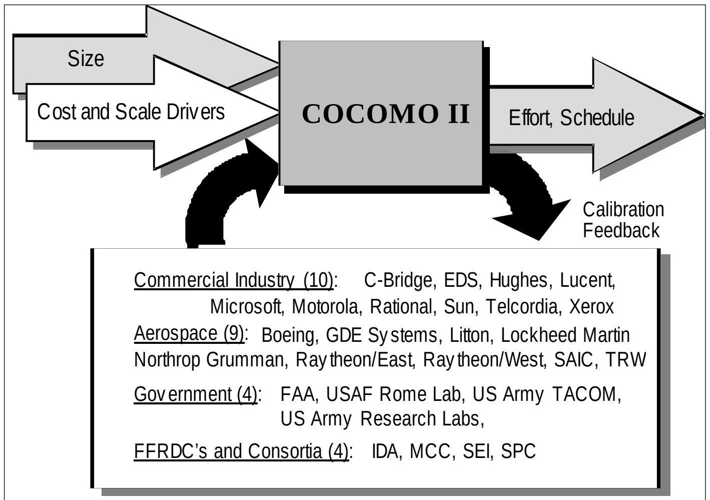
Xerox is interested in the application of COCOMO for software cost estimation, and participates in the University of Southern California/Center for Software Engineering (USC/CSE) Industrial Affiliates Program. At this time 27 industrial affiliates provided data or input to enhance and fine-tune the COCOMO II model (as seen in Figure 4).

Here we provide a conceptual introduction to COCOMO. For up-to-date details, see the appropriate materials on the web site, http://sunset.usc.edu, or in hard copy format [12] for an introduction. (Please note that since publication of that article, the model was renamed to COCOMO II from COCOMO 2.0.)

The COCOMO II model uses 161 data points from affiliates' databases, and is the enhanced version of the earlier COCOMO 81, which was developed using only 64 data points from TRW. Besides refining the model, the university also provides MS/Windows, Sun, and Java versions of the tool, based

Risk Class and Element from Taxonomy Based Questionnaire	Issues and Concerns Recorded during the SRE sessions	Risk Magnitude Rating			
		A	B	C	Team
Program Constraints/Resources	Currrent Plan is schedule driven	6	9	9	8.0
Program Constraints/Resources	Bottom-up plans do not support the schedule	6	9	9	8.0
Development Environment/Management Process	Management is not ready to reconcile the differences between the engineering plan and the business plan	9	6	9	8.0
Program Constraints/Resources	Top-level plan is unrealistic, and it is not based on past track-record and experience	6	6	9	7.0
Development Environment/Management Process	Inability in estimating effort due to the lack of experience with the new technology	4	6	9	6.3
Development Environment/Development Process	Feedback from implementers to architects takes too long, and there is no closure on certain issues	4	9	6	6.3
Development Environment/Development System	Capacity limitaions of the development system (network bandwidth and the speed of compilation) impact schedule	6	6	6	6.0
Program Constraints/Resources	Lack of confidence in the current plans	4	6	6	4.6
Development Environment/Development System	Lack of availibility of adequate number of software licenses	3	4	4	3.6

Figure 3. *Sample Record of Risk Issues during an SRE*


Size
Cost and Scale Drivers
COCOMO II
Effort, Schedule
Calibration Feedback
Commercial Industry (10): C-Bridge, EDS, Hughes, Lucent, Microsoft, Motorola, Rational, Sun, Telcordia, Xerox
Aerospace (9): Boeing, GDE Systems, Litton, Lockheed Martin Northrop Grumman, Raytheon/East, Raytheon/West, SAIC, TRW
Government (4): FAA, USAF Rome Lab, US Army TACOM, US Army Research Labs,
FFRDC's and Consortia (4): IDA, MCC, SEI, SPC

Cocomo II Cost and Scaling Drivers

Cost Drivers	
ACAP	Analyst Capability
AEXP	Applications Experience
CPLX	Product Complexity
DATA	Database Size
DOCU	Documentation to match lifecycle needs
LTEX	Language and Tool Experience
PCAP	Programmer Capability
PCON	Personnel continuity
PEXP	Platform Experience
PVOL	Platform Volatility
RELY	Required Software Reliability
RUSE	Required Reusability
SCED	Required Development Schedule
SITE	Multi-site operation
STOR	Main Storage Constraint
TIME	Execution Time Constraint
TOOL	Use of Software Tools

Scaling Drivers	
PREC	Precedentedness
FLEX	Development Flexibility
RESL	Architecture/Risk Resolution
TEAM	Team Cohesion
PMAT	Process Maturity

Figure 4. *COCOMO II Software Cost Estimation Model*

Actual Rating	Entry for the tool's screen	Very Low	Low	**Nominal**	High	Very High
Rating Guidelines	Relationship to nominal	75%	85%	**100%**	130%	160%

Figure 5. *COCOMO Rating Guidelines for* ***SCED***

SEVERITY	PROBABILITY		
	Improbable	Probable	Very Likely
Catastrophic	3	6	9
Critical	2	4	6
Marginal	1	2	3

Figure 6. *Risk Magnitude Matrix*

on the current version of the model. Due to the model's popularity, a number of industrial tool vendors incorporated the COCOMO II model into their software cost estimation tool offerings.

Size is the main driver of cost, so the first step of cost estimation is to provide proper size estimation for the project. COCOMO II accepts source line of code (SLOC) and function point input. During the estimation process, the estimator determines the value of scaling constants and cost drivers, using the supplied rating tables, and enters the value in the appropriate screen of the tool. An example based on the COCOMO Model Definition Manual for the cost driver rating guideline is shown in Figure 5.

SCED (required development schedule) belongs to the group of cost drivers that characterize the project to be estimated, and it measures the schedule constraint imposed on the project team. The ratings are defined in terms of percentage of schedule stretch or compression with respect to a nominal schedule for a project requiring a given amount of effort. Compressed or accelerated schedules tend to produce more effort in later phases of development because more issues are left to be determined and resolved.

"Hard" and "Soft" Approaches

As indicated previously, we classify the different risk analysis approaches based on the mathematical rigor required. The first example of a hard approach is offered by Madachy, where he creates the risk taxonomy outright, around the COCOMO cost drivers. This impressive system uses knowledge-engineering methods to refine the risk quantification scheme [7].

In the second example, Känsälä uses Madachy's basic approach, but instead of working around a particular cost estimation tool, he derives his own risk database using risk questionnaires and historical project data. Experimental integration of this risk front-end, *RiskMethod*, was carried out with three different cost estimation tools [8].

The underlying principle in both cases is the use of regression analysis to determine the model's internal coefficients. The approaches require extensive calibration to achieve acceptable results. Finally, the authors' initial objective was not only to assess and prioritize, but also to quantify software risks.

Känsälä also provides the first example of a soft approach. In this TRW approach, the author defines a list of specialized risk issues that could be viewed as a one-level risk taxonomy. This taxonomy is used by internal risk review boards to assess risks via intensive monthly sessions with key representatives of functional and support areas. No particular efforts are made to quantify the impact of identified risks.

Conrad and Shishido give a second example, in which a more complex taxonomy is used by a combined external-internal assessment team in a single assessment. For several days the team formally interviews a cross-section of the development organization for several days. The interviewees are not necessarily key representatives, and they represent a vertical sample of people in the development organization. Nonattribution is a key guiding principle during the sessions, and the names of those raising concerns are kept confidential [9].

	Risk Class and Element from Taxonomy Based Questionnaire	Issues and Concerns Recorded during the SRE sessions	COCOMO Driver Mapping	Team Risk Magnitude Rating
1	Program Constraints/Resources	Current plan is schedule driven	SCED	8.0
2	Program Constraints/Resources	Bottom-up plans do not support the schedule	SCED	8.0
3	Development Environment/ Management Process	Management is not ready to reconcile the differences between the engineering plan and the business plan	SCED	8.0
4	Program Constraints/Resources	Top-level plan is unrealistic, and it is not based on past track record and experience	SCED	7.0
5	Development Environment/ Management Process	Inability in estimating effort due to the lack of experience with the new technology	SCED	6.3
6	Program Constraints/Resources	Lack of confidence in the current plans	SCED	4.6
		Subtotal for	SCED	41.9
		Average for	SCED	6.5

Figure 7. *Worksheet to facilitate mapping*

The common theme in both cases is the use of interviews and guided discussions, with no provisions for risk quantification.

Xerox used the SEI approach, and the following two, main advantages were identified:

- The taxonomy was broad, but also proved to be detailed enough to carry out efficient interviews.
- The nonattributional approach was useful in uncovering well-known risks obvious to the developer community, but due to lack of trust or broken communication, unacknowledged by management.

It had become obvious that the ability to quantify is helpful in prioritizing and presenting the risks to the decision authority.

These experiences led us to recognize that it would be useful to combine the best of both worlds: keep the SEI method as a risk front-end and use COCOMO for quantification. The benefit is that we maintain flexibility and easy customization at the front-end, while using an already calibrated COCOMO model to quantify the results. We did not perceive the benefit of an expert-system approach, because it introduced a complicated and, in our opinion, unnecessary calibration and learning process.

Using COCOMO II To Quantify Software Risk Evaluation Results

On a conceptual level, the task can be phrased as a m:n mapping from the risk taxonomy into the COCOMO scale and cost-driver taxonomy. This complexity resulted in indentifying nearly 600 risk conditions in Madachy's tool. While the precise mapping and the identification of all possible combinations are necessary to create a knowledge-management tool, we found that the goal is never fully accomplished, and customization has to take place before use anyway. The steps of the recommended soft approach are:

1. Carry out an SRE. Do this by using the detailed guidelines of Sisti et. al. [10] To prepare for sessions, the assessment team has to customize the TBQ before and, to some extent, during the interviews to concentrate on the appropriate TBQ subset. An example for customization is shown in Figure 2. *Impact of hardware/software partitioning* was added to the standard TBQ questionnaire to accommodate special Xerox requirements.
2. Map risks into COCOMO. Mark the relevant COCOMO drivers on the worksheets. Figure 7 shows the relevant fragment of the sample risk record, mapped to the SCED cost driver. All the shaded rows on Figure 3 are removed, since they do not map into SCED.
3. Determine baseline estimates. Execute COCOMO estimation with baselined scale and cost factors. For example, COCOMO II would give 16.8 person/month effort estimation for a 5000 SLOC program, where for sake of simplicity nominal values were used for all cost and scale drivers. Please note that this baseline reflects typical, but hypothetical, project conditions fitting the average profile of COCOMO industrial affiliates. For more accurate estimations, the model has to be calibrated with local organizational data, and the drivers have to be set according to the organization's own, original planning conditions.
4. Determine risk-adjusted estimates and compare results. The main objective is to execute COCOMO estimation with risk-adjusted scale and cost factors and determine the difference between baselined and risk-adjusted estimates. In the simple case we present, this means determing the model's sensitivity to SCED cost-driver changes. Figure 8 shows mapping of the risk magnitude into the cost-driver rating.

Note that the mapping is not automatic, and done for all drivers separately, based on their specifics. In case of SCED, for example, the issue from risk identification point of view is forced schedule compression, so there is no point in analyzing effects of a high or very high rating representing a relaxed, instead of compressed, schedule. Applying the risk adjusted value of very low for the SCED cost-driver, COCOMO II at this time would give 21.6 person/month effort estimation for the same 5000 SLOC program. Again, note that all other cost and scale drivers are set to nominal. The demonstrated what-if scenario shows, that in case of a 5000 SLOC program, the impact of a roughly 75-85 percent schedule compression would be a 28 percent increase in effort.

Summary

Risk management is one of the most critical and most difficult aspects of software project management. It is evident, that SEI/SRE as a risk identification method yielded better, more detailed, and more relevant risk items than any input processes

Risk Magnitude from SRE	1	2	3	4	5	6	7	8	9
SCED Rating for COCOMO	NOMINAL		LOW			VERY LOW			

Figure 8. *Mapping Risk Magnitude into Cost Driver Rating*

customarily used with hard risk assessment tools. This is a preferred lightweight approach, because it uses established, familiar, and well-tested tools. Customization and calibration are always needed, even when comprehensive and sophisticated knowledge-engineering-based tools are used. Therefore, we conclude that applying customization effort to existing tools in a lightweight setup is a more efficient approach than purchasing and implementing new, complex tools.◆

References

1. Voldese, I. S. Risk Analysis Using Parametric Cost Models, *Proceedings of the ISPA/SCEA First Joint International Conference*, Toronto, Ontario, Canada, June 16-19, 1998, pg. 1382-1409.
2. Boehm, Barry W. Software Risk Management. *IEEE Computer Press*, 1989.
3. *Taxonomy-Based Risk Identification*. Technical Report CMU/SEI-93-TR-16.
4. Garvey, P.R. A Risk Management Information System Concept for Systems Engineering Applications. Leesburg, Va.. *28th Annual Department of Defense Cost Analysis Symposium*, September 1994.
5. Moynihan, T. Inventory of Personal Constructs for Risk Researchers. *Journal of Information Technology*, Vol. 6, No. 4, 1996.
6. Barki, H., Rivard, S., and Talbot, J. Toward an Assessment of Software Development Risk. *Journal of Management Information Systems*, Vol. 10, No. 2, 1993.
7. Madachy, R.J. Heuristic Risk Assessment Using Cost Factors. *IEEE Software*, May/June 1997.
8. Känsälä, K. Integrating Risk Assessment with Cost Estimation. *IEEE Software*, May/June 1997.
9. Conrow, E.H., and Shisido. P.S. Implementing Risk Management on Software Intensive Projects. *IEEE Software*, May/June 1997.
10. Sisti, F.J., Joseph, S. *Software Risk Evaluation Method, Version 1.0*. Technical Report CMU/SEI-94-TR-19.
11. Hantos, Peter. Experiences with the SEI Risk Evaluation Method. Portland, Oregon. Pacific Northwest Software Quality Conference, 1996.
12. Boehm, Barry W., Clark, B., Horowitz, E., Westland, C., Madachy, R., and Selby, R. COCOMO 2.0 Software Cost Estimation Model. *American Programmer*, July 1996.

Notes

1. "Covert" taxonomy means that the risk sources and their structure are not visible, and they are embedded in the tool. In case of "overt" taxonomy there is an explicit reference to the description of the risk hierarchy.
2. From *Webster's Dictionary*: tax-on-o-my (tak-'sän—me) *n.* 1: the study of general principles in scientific classification.

About the Author

Peter Hantos is department manager at Xerox, in charge of Software Quality Assurance, SPI, Product Quality Assurance, and Reliability. Previously, he was principal scientist of the Xerox Corporate Software Engineering Center, where he developed the corporate software development standard and the software technology readiness processes. He holds a master's and a doctorate degree in electrical engineering from the Technical University of Budapest, Hungary. Dr. Hantos is a senior member of the Institute of the Electrical and Electronics Engineers, and a member of ACM.

Xerox Corporation
701 South Aviation Blvd., MS: ESAE-375
El Segundo, Calif. 90245
Phone: 310-333-9038
Fax: 310-333-8409
E-mail: peter.hantos@usa.xerox.com

Appendix SEI Software Risk Taxonomy

A. PRODUCT ENGINEERING	B. DEVELOPMENT ENV.	C. PROGRAM CONSTRAINTS
1. **Requirements**	1. **Development Process**	1. **Resources**
a. Stability	a. Formality	a. Staff
b. Completeness	b. Suitability	b. Budget
c. Clarity	c. Process Control	c. Schedule
d. Validity	d. Familiarity	d. Facilities
e. Feasibility	e. Product Control	
f. Precendented		
g. Scale		
2. **Design and**	2. **Development System**	2. **Contract**
a. Functionality	a. Capacity	a. Type of contract
b. Difficulty	b. Suitability	b. Restrictions
c. Interfaces	c. Usability	c. Dependencies
d. Performance	d. Familiarity	
e. Testability	e. Reliability	
f. Hardware Constraints	f. System Support	
g. Non-Development Software	g. Deliverability	
3. **Code and Unit Test**	3. **Management Process**	3. **Program Interfaces**
a. Feasibility	a. Planning	a. Customer
b. Testing	b. Project Organization	b. Associate Contractors
c. Coding/Implementation	c. Management Experience	c. Subcontractors
	d. Program Interfaces	d. Prime Contractor
		e. Corporate Management
		f. Vendors
		g. Politics
4. **Integration and Test**	4. **Management Methods**	
a. Environment	a. Monitoring	
b. Product	b. Personnel Management	
c. System	c. Quality Assurance	
	d. Configuration Management	
5. **Engineering Specialties**	5. **Work Environment**	
a. Maintainability	a. Quality Attitude	
b. Reliability	b. Cooperation	
c. Safety	c. Communication	
d. Security	d. Morale	
e. Human Factors		

12th Annual Software Technology Conference

"Software and Systems—Managing Risk, Complexity, Compatibility and Change" is the theme for the Software Technology Conference (STC 2000), set for April 30-May 5 in Salt Lake City, Utah.

As we sit on the cusp of a new millennium, the Information Technology (IT) industry offers a wider and more dynamic range of IT solutions than ever before. Compounding the software professional's dilemma is the speed at which evolutionary as well as revolutionary change occurs. In what direction do we go? Which technology to buy? Can we facilitate the flow of information and still maintain security? No one possesses unlimited budgets or has a crystal ball; making the right choice the first time will be difficult but essential to the success of your enterprise. You, the developers, acquirers, integrators, and users of software are essential to the creation of a successful future. Only through a collective exploration can it be achieved. STC 2000 will provide an excellent forum to discuss, contemplate, and explore the myriad of solutions that our industry has to offer.

Sponsors, General Session

The U.S. Department of the Air Force, Department of the Army, Department of the Navy, and the Defense Information Systems Agency (DISA), have again joined forces to co-sponsor STC 2000, the premier Software Technology Conference in the Department of Defense. Once again Utah State University Extension is the conference nonfederal co-sponsor. We anticipate more than 3,500 participants from the military services, government agencies, contractors, industry, and academia.

The government co-sponsors are Lt. Gen. David J. Kelley, Director, DISA; Lt. Gen. William Campbell, Director of Information Systems for Command, Control, Communications, and Computers, U.S. Army; Dr. Donald Daniel, Deputy Assistant Secretary of the Air Force for Science, Technology, and Engineering, U.S. Air Force; and Rear Adm. Kenneth D. Slaght, Chief Engineer, Space and Naval Warfare Systems, U.S. Navy.

The general session will be held May 1. Speakers will include high-level officials representing government, industry, and academia. In addition, the co-sponsors will host a question-and-answer general session on May 2. Conference attendees are encouraged to turn in their questions to conference management prior to the conference or to the on-site control room on May 1. This is a great chance to get answers from our senior leaders on important issues.

Special Sessions

Sponsored presentation tracks will be provided by DISA, Real-time Defense Information Infrastructure Common Operating Environment (DII COE), Software Productivity Consortium, Data & Analysis Center for Software, Army Cost Estimation, Institute of Electrical and Electronics Engineers (IEEE), International Conference of Software Engineering, National Security Agency (NSA) & DISA-Intel Classified, Office of the Secretary of Defense-Software Collaboration, and the Software Technology Support Center.

A special classified intelligence meeting, in conjunction with the conference, is planned for May 3. Top secret Special Intelligence/TK clearances will be necessary for this one-day track. Details are in the registration brochure.

Side Meetings, Workshop

One of the biggest benefits of STC is the great networking opportunity. Side meetings and birds-of-a-feather meetings will be available. If you are interested in reserving a time for one of these meetings, please call us at 801-777-9828. We will be glad to schedule a meeting for you.

Take advantage of IEEE's one-day workshop on May 5, Guide to the Software Engineering Body of Knowledge (SWE-BOK): Overview and Applications. This guide should be useful in developing software-engineering curricula, accrediting curricula, licensing examinations, and describing and certifying competencies. The workshop is included in your conference fee.

On the Agenda

Presentation tracks include, but are not limited to:

- Capability Maturity—Models, Assessments, Evaluations
- Capability Maturity Model Integration®
- Collaborative Engineering
- Defense Information Infrastructure Common Operating Environment (DII COE)
- Distributed Computing
- Education and Training
- Electronic Commerce
- Emerging Technologies
- Internet/Intranet
- Interoperability
- Knowledge Management
- Measurement
- Modeling and Simulation
- Network Centric Systems
- Object-Oriented Technology and Languages
- Open Systems and Architectures
- Process Improvement
- Project Management
- Quality Assurance
- Real-Time DII COE
- Re-engineering
- Risk Management
- Software Acquisition
- Software Architecture
- Software Cost Estimation
- Software Implementation
- Software Policies and Standards
- Software Testing
- System Requirements
- Total Cost of Ownership

STC 2000 Exhibiting Organizations (As of 11/29/99)

Abelia Corp.
AccessData Corp.
Ada Core Technologies Inc.
Aonix
As-One Inc.
AXENT Technologies Inc.
BMC Software Inc.
Computer Sciences Corp.
Concurrent Computer Corp.
DDC-I
Defense Acquisition Deskbook
Defense Information Systems Agency
Document Automation & Production
EDS
Engineering Systems Solutions
Enterworks Inc.
Federal Data Corp.
Forte Software Inc.
General Services Administration Federal Technology Service
Gensym Corp.
GRC International Inc.
Green Hills Software Inc.
IBM Corp.
IIT Research Institute
Information Resources Management College
Informix Software Inc.
Integrated Chipware
Integrated Computer Engineering Inc.
Integrated System Diagnostics Inc.
International Function Point Users Group
JOVIAL ITS Program Office
Logicon Inc.
Lotus Development Corp.
Management Concepts Inc.
McCabe & Associates Inc.
MERANT
Micron Government Computer Systems Inc.
Microsoft Corp.
Mobius Management Systems Inc.
National Security Agency
Northrop Grumman Corp.
Novell
Objective Interface Systems Inc.
Onix Networking
OO-ALC/TIST
Oracle Corp.
PEO STAMIS
PeopleSoft Inc.
pragma Systems Corp.
Quality Checked Software
Quality Systems & Software
Rational Software
Real-Time Innovations Inc.
RS Information Systems Inc.
Science Applications International Corp.
Software Configuration Solutions Inc.
Software Productivity Consortium
Software Technology Support Center
SPAWAR
Sun Microsystems
Sybase Inc.
Template Software Inc.
TeraQuest Metrics Inc.
The MathWorks
TimeSys Corp.
Tivoli Systems Inc.
TSI Software
United Defense
U.S. Air Force
U.S. Army
U.S. Army Information Systems Engineering Command
U.S. Army CECOM, Software Engineering Center
Utah State University Extension
Vitech Corp.
Weapon Systems Technical Working Architecture Working Group
WR-ALC/LYS Software Engineering Div.
Z Microsystems Inc.

Reservations, Registration, Exhibit Booths

The Salt Lake Convention and Visitors Bureau (SLCVB) is taking hotel room reservations. To reserve a room, fill out the form in the registration brochure and fax it to the SLCVB Housing Bureau at 801-355-0250. There will be buses during conference hours for transportation between the Salt Palace and city center hotels. Government-rate rooms go quickly, so do not delay.

The conference fee structure:

Discounted registration fee paid by April 3, 2000

Active Duty Military/Government	$540*
Business/Industry/Other	$665

Regular registration fee paid after April 3, 2000

Active Duty Military/Government	$600*
Business/Industry/Other	$750

*Military rank (active duty) or government GS rating or equivalent is required to qualify for these rates.

The official STC 2000 registration brochure was mailed early last month. Send in your registration forms, with your credit card number now, and it will not be charged until April 10. You do not have to wait until May to register.

Registration and the layout of the exhibition area can be found in our brochure, which is available on the Internet at http://www.stc-online.org. Vendor information also is available at this address. Space rental rate is $1,375 per 10-foot by 10-foot booth. Late registration received after Feb. 18, should space be available, will rent for $1,575. Please write to us at stcexhibits@ext.usu.edu, or call 435-797-0047 for more information about the exhibition.

Contact Us

If this issue of CrossTalk was mailed to you, you are on our mailing list. If you had to borrow a copy, please contact us to be added to our mailing list. You may use our web site at http://www.stc-online.org/ for further information about STC.

Dana Dovenbarger, Conference Manager
Lynne Wade, Assistant Conference Manager
Software Technology Support Center
OO-ALC/TISEA
7278 Fourth St.
Hill AFB, Utah 84056-5205
Voice: 801-777-7411 DSN 777-7411
Voice: 801-777-9828 DSN 777-9828
Fax: 801-775-4932 DSN 775-4932
E-mail: dana.dovenbarger@hill.af.mil
lynne.wade@hill.af.mil

Salt Palace Convention Center, Site of STC 2000

Risk Management Rollout and Installation at the NRO

Acquisition reform provides government program managers and their contractors more opportunities than ever before, but it also greatly increases risk. It is no surprise that Department of Defense (DoD) organizations face reduced resources, deplenishment of critical skills, increased congressional and public scrutiny, and a demand for streamlined operations and quality program results. The need for risk management is becoming apparent. But how? When program managers are under pressure to meet accelerated schedules and within budget, can risk management be integrated into already-burdened programs? This article shows how the National Reconnaissance Office (NRO), an organization with a dual DoD and security mission, introduced a disciplined risk management process that has provided significant cost-benefits.

DoD organizations increasingly operate as partnerships: government program managers and contractors are responsible for producing sophisticated and complex systems through integration of capabilities to deliver program results. The pressures that affect the DoD community also impact the NRO. This organization experiences the additional challenges of meeting mission requirements that integrate DoD and security missions. The organization faces a combination of new technologies, system complexity, tight schedules, and often-unstable requirements. With increased congressional oversight and public awareness, the NRO is experiencing pressure to deliver more complex and accelerated security programs than ever before.

In December 1995, the Undersecretary of Defense, Acquisition and Technology issued a memorandum, *Reducing Life Cycle Costs for New and Fielded Systems*. It acknowledged that "There are risks to be taken and risks to be avoided. When risks are taken, we will put in place appropriate risk management and contingency plans" [1]. The most recent guidance, released in May 1999, *Risk Management Guide for DoD Acquisition*, strengthens this policy. Risk management is defined as "an integral component of policy and strategy to develop and field systems responsive to user needs" [2].

NRO leadership has also directed that risk management is a critical element for program success. In fact, NRO Directive 7 of policy document *NRO Acquisition Management* emphasizes:

"Effective acquisition planning and aggressive risk management by both government and industry are essential for success. Program decisions and resource commitments must be based on consideration of executable options and plans for, and progress in, controlling risk" [3].

The *Catch-22* in today's context is that, while risk management is more necessary than ever, finding time and resources to install a sound program creates a need to prove real and lasting value. As many who have tried to install risk management in their programs know, expanded awareness of its importance, and guidance on desired results, do not necessarily translate into effective, interactive processes that provide desired results.

For the past two years, the imagery intelligence (IMINT) organization at NRO, in partnership with the Software Engineering Institute (SEI), created and installed a risk management process that became an integrated aspect of program operations. The full story of this risk management initiative is available in *Rollout and Installation of Risk Management at the IMINT Directorate, National Reconnaissance Office*, a December 1999 technical report published by SEI [4]. Three critical success factors in this program installation were:

1. Visible and committed sponsorship.
2. Culture change that supported open communication for the surfacing and mitigation of risks.
3. A disciplined, forward-looking, continuous risk management, infrastructure, and rational, well-thought-out installation process.

These three critical success factors are interlocking and mutually reinforcing. Each of these was present and engaged at NRO, as described below.

Committed, Visible Sponsorship

"Leaders do not have a choice about whether to communicate," says Edgar Schein, professor at the MIT Sloan School of Management. "Leaders send messages whether they wish to or not. People in organizations are constantly looking to their leaders for cues about what is acceptable behavior. And it is not merely public statements that people hear and believe; it is the entire range of messages sent through behaviors and their consequences, organizational mechanisms, and events that have impact" [5].

The original pilot Risk Management program was undertaken by the Command and Control Division (CCD) at IMINT. This was the first NRO program to engage in a full-scale risk management program [6].

This pilot project was not without its challenges. The initial reaction by CCD participants was a measure of skepticism coupled with clear expression of their expectations.

"We were not interested in just learning a new vocabulary," said one area manager, "if risk management did not help us get our jobs done every day, then we were not interested. Risk management needs to show value by providing cost-efficiences, better scheduling, and technological capability."

The CCD division chief, and his area manager that he charged with risk management, decided that their best opportunity to achieve results with risk management would require a unified, coherent, disciplined process that involved all division staff. Accordingly, the division held a risk clinic to create its focused process. Once the decision was made to build a risk management infrastructure and process, the CCD division chief decided it was important that he be present at all critical meetings. The division chief integrated risk management into technical assessments, program reviews, and the monthly joint government/contractor meetings to manage joint risks, the Team Risk Reviews (TRRs).

As a result, the government program managers and contractor managers created a dialogue that, over time, opened communication to a level of candor that provided technical, schedule, and cost mitigations. An example of an early success resulted from identifying a risk in the methodology by which the program planned to manage its Operations and Maintenance (O&M) process. Up to this point the O&M activities and the development activities were separated into different organizational elements under different contracts.

Due in part to this segregation, the processes were inherently expensive and made it easy for deplenishments in functionality to occur. The risk mitigation strategy was development of the Integrated Development and Maintenance Organization (IDMO). The IDMO absorbed the maintenance functions traditionally managed by the operational site and integrated them into the development organization. The goal was to gain synergy through a single reduced staff that would manage a consolidated maintenance and development effort.

The sponsorship factor is undeniable: both government and contractor managers exhibited clear support for necessary changes to achieve constructive outcomes.

Once the pilot program had proved the value and efficacy of risk management to the satisfaction of this division and to Al Krum, the system program director, he decided to install risk management at the system level.

Recognizing the importance of focus on the entire program, he confirmed that, for risk management to be successful, "Management commitment is invaluable. Managers cannot assign risk management leadership to individual contributors; risk management will not be taken seriously without appropriate and visible leadership" [4].

As evidence of his commitment to risk management throughout his system, Krum launched the Executive System-level Risk Management Team (ESRT), that included all division chiefs, to lead the way for risk management by other divisions across the system. Next, he directed that each division would undergo training in risk management processes and procedures, to develop a common language and set of tools to use within divisions and across the system.

In addition to authorizing a rollout and installation process across the organization, Krum "walked his talk." During development of the risk management process, division chiefs continued to question the durability of sponsorship for risk management. Although Krum had clearly stated his sponsorship for risk management, there were mixed views regarding management's seriousness about the initiative. To reinforce his commitment, he used various forums to present his vision for the risk management effort and his expectations for coherent, consistent communications up the chain to management. The result was strengthened support of risk management by most division chiefs, as well as evolving practices within the divisions.

A risk management infrastructure is essential to support the kinds of communication necessary for effective risk management.

To further clarify the seriousness of risk management at the system level, the program manager led the division chiefs to define the level of risks appropriate for system-level discussion. Accordingly, the ESRT defined its criteria for system-level risks as follows:

- Impact on program commitments.
- Risks deemed as priority due to being on the program's critical path or on any division's critical path.
- Exceeded planned schedule slack, and those that resulted in a negative margin (seriously eroded management reserve).
- Cost impact exceeds planned budget.
- Impact on program interfaces, internal or external to the program.

System risk management allowed the program, for the first time, to analyze and work together on interdependencies at the system level. Through the division risk management processes and the ESRT, a consistent system process was installed. As divisions installed their own process, they were increasingly able to communicate mission-critical risk management concerns to their contractors.

Culture Change to Support Open Communication, Risk Mitigation

The 1999 DoD *Risk Management Guide* emphasizes a risk management approach that is disciplined, forward looking, and continuous [2]. "Our goal," program director Krum said in describing IMINT's aim of risk management, "was to build a system in which people would think ahead, mitigate risks, and reduce the likelihood of system delays, depletion of management reserve, and system failures.

"To accomplish this, we knew that risk management would require a culture change to one where people would openly discuss those very areas that are most likely to be uncomfortable. We wanted to build program success on a platform where leaders and contributors *put their cards on the table*" [4].

NRO had undergone a number of studies that analyzed specific cultural barriers to effective program functioning. In particular, a study by Malcolm Baldridge in 1996 pointed to a need for NRO to move from a culture of risk avoidance to the kind of open communications that elicit widespread knowledge and information sharing [7]. Risk management, in particular, requires early and open exchange of key information to allow for timely mitigation. Several studies have shown that, in the great majority of program failures, one or more key technical project staff knew in advance there was a serious risk of failure.

The forums established at IMINT provided for this kind of discussion. Krum said, "Risk management forums at both system and divisional levels are not so much a place where you 'don't shoot the messenger' as one where 'there is no messenger to shoot because there is not a crisis yet.'" [4]

In the example cited above, the advent of an IDMO organization was not readily embraced by the O&M organization. O&M members thought that it took away some of their flexibility to utilize level-of-effort resources to address the new approach, and required a scheduling discipline that was contrary to the organization's existing business practices. In addition, the maintenance budget would be turned over to development.

Open-forum discussions established in the monthly team risk review meetings allowed identification of a potential budgetary risk in the newly configured approach. The risk was that the original O&M program might not have budgeted sufficient resources to support new architecture being delivered.

As details of the risk were developed and discussed openly between government and contractor, it turned out that there was a budget shortfall. With this early identification, the management team could provide a budget wedge and secure necessary funding to acquire key resources and meet availability requirements.

To arrive at this level of candor and critical information exchange, leaders set the tone. Contributors and others must accept accountability to support the effort. A risk management infrastructure is essential to support the kinds of communication necessary for effective risk management.

Of course, all government organization, including NRO, have grown change-weary. To foster acceptance of risk management, a building of trust in the process was necessary, so that it would not be seen as the *change process du jour.* The risk management infrastructure supported the growth of trust in the process.

Risk Management Process and Infrastructure

To leverage a lasting risk management process across a complex system, where the pressure of current crises can erode the best intentions, a sturdy yet streamlined infrastructure is essential. For new initiatives that create a culture change, building a solid, well-designed infrastructure to support the change may initially seem burdensome. However, the infrastructure, if well-planned, can become a support for ultimate efficiencies and integration across system program management. The infrastructure further supports an alert, continuous process of risk management watchful for emerging and changing risks over a program life cycle.

A key driver behind the installation process at IMINT was to support division leaders who could provide sponsorship, model risk management behaviors, and act as in-house mentors for the process over time. Another critical element was to install a smoothly functioning operational infrastructure (teams, processes, practices, and information resources) to leverage continuous risk management and improvement of the process.

The infrastructure installed at IMINT included the following:

- Conducting software risk evaluations with government and contractors for the pilot program to identify initial program risks and plan mitigation strategies.
- Establishing divisional risk management practices, which include regular forums where risks are identified, planned, tracked, and controlled—in staff meetings, program review sessions, or specified risk reviews.
- In some cases, establishing goverment/contractor integrated product and process team-type TRRs for monthly identification, planning, tracking, and control of joint risks.
- Establishing the system-level risk management team, where division chiefs and the program director or his deputy meet monthly to discuss and monitor system-level risks.
- Designing a communications architecture that provides clear guidelines to all staff members on roles and mechanisms for communicating and working risks.
- Developing consistent, system-wide risk information documents, tracking charts, and a custom-tailored risk management tool initially designed by the CCD pilot program, eventually leveraged for system-wide use.

These streamlined infrastructure elements supported the risk management process at IMINT, and in turn are increasingly supported by the expanding risk management community. The contributors at IMINT found a return on investment in risk management that ranged from such subtle changes as “awareness has increased; we no longer just look at today’s problems” to major crisis-averting risk management. IMINT still bears fruit.

As a final example, a mission-critical risk identified early on during one of the original IMINT risk assessments in 1997 was a portfolio of related risks involving scheduling and specifics of system-level testing, which were key to successful program delivery. In response to this major risk, an integrated set of mitigation strategies were translated into a mitigation plan. The mitigation plan received clear guidance from the government sponsor, with collaborative input from government and contractor TRR team members. With close monitoring at the monthly TRRs and ESRT sessions, and with instantiation of several contingency actions, the program delivery was not only on time, but successful in all defined aspects.

Conclusion

While risk management program integration is almost never easy, as an IMINT contractor said, “It is important to do the hard right thing than the wrong easy thing.” By applying the three key success factors outlined above—committed sponsorship, a culture moving toward more open communication, and a reliable infrastructure to support continuous risk management—IMINT achieved successful risk management results in support of mission-critical programs into the new century.◆

References

1. *Reducing Life Cycle Costs for New and Fielded Systems, Undersecretary of Defense (Acquisition and Technology) Memorandum*, December 1995.
2. *Risk Management Guide for DoD Acquisition, DoD Test, Systems Engineering and Evaluation, Defense Acquisition University, and Defense Systems Management College,* Defense Systems Management College Press, Fort Belvoir, Va., May 1999.
3. *NRO Acquisition Management Directive 7* (NROD 82-2, OPR (P&A), National Reconnaissance Office, Aug. 6, 1997. Available on the web at http://www.dsmc.dsm.mil/pubs/gdbks/risk_management.html
4. Loveland Link, Jo Lee, Rick Barbour, Al Krum, and August C. Neitzel, *Rollout and Installation of Risk Management at the IMINT Directorate, National Reconnaissance Office,* CMU/SEI-99-TR-009, ESC-TR-99-009, SEI Technical Report, December 1999.
5. Schein, Edgar H., *Organizational Culture and Leadership,* Jossey-Bass Business and Management Series, San Francisco, 1997.
6. Neitzel, August C., Jr., *Managing Risk Management,* CrossTalk: July 1999.
7. *IMINT Malcolm Baldridge National Quality Award Assessment Consolidation Report,* National Reconnaissance Office, May 1996.

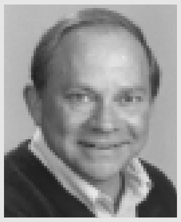
About the Authors

August Neitzel is Director, EIS Ground Group, responsible for system delivery. During the period of risk management rollout and installation, he was division chief of IMINT's command and control acquisition effort. In this capacity, he led the pilot program for the initial IMINT risk management initiative. In addition, he served as the contracting officer's technical representative for the command and control acquisition contract. Neitzel joined the CIA in 1975. In 1982, he began working for the NRO. His career there has spanned the SIGINT program and virtually all aspects of the IMINT program. Neitzel earned a master's degree in electrical engineering from Drexel University after completing a tour of duty with the Air Force. He is a member of Eta Kappa Nu and the Institute of Electrical and Electronic Engineers. He is certified as a Level III COTR, and received the CIA Intelligence Commendation Medal.

NRO
4101 Pleasant Valley Road
Chantilly, Va. 20151
Voice: 703-808-2038

Jo Lee Loveland Link is a visiting scientist at SEI, with more than 20 years of providing guidance for strategic direction of government, military, and private sector organizations. For the past six years, she has worked with technical programs to establish strategy and infrastructure for process improvement and risk management initiatives. With Richard Barbour, she provided risk management rollout and installation services across this IMINT organization. She has participated on Capability and Maturity Model®-based assessment teams and teaches several SEI workshops, including Managing Technological Change, Consulting Skills, and customer-tailored Risk Management. A certified Senior Organization Development and Change Specialist with post-graduate work in applied behavior science, Loveland Link has a bachelor's degree in organization behavior/adult education, and is author of more than 30 publications on strategic planning, management, and corporate culture.

Software Engineering Institute
4301 Wilson Blvd., Suite 910
Arlington, Va. 22203
Voice: 703-709-9217 or 703-908-8232
Fax: 703-904-8330
E-mail: jll@sei.cmu.edu



Richard E. Barbour is a senior member of the technical staff at CISE. For the past year, he has provided software capability evaluations (SCEs) to international software companies. In 1998-99, Barbour was instrumental in SCE assessments for the NRO future imagery architecture program. Prior to joining CISE, he was in the SEI software engineering process management program as the acquisition improvement project leader, serving as lead for the SEI-NRO risk management initiative. He had been in the SEI Process Program, developing and implementing Capability Maturity Model®-based appraisals for internal process improvement and software capability evaluations. He also spent a year with the SEI Transition Partner, Integrated System Diagnostics Inc., developing the SCE v. 3.0 method. Barbour has more than 24 years of experience in managing and acquiring software systems, and is a retired Navy Commander from the antisubmarine warfare, patrol squadrons (P-3) community. His last assignment was as deputy program manager for the Next Generation Computer Resources program with the Space and Naval Warfare Systems Command. He also was deputy program manager for the Software Technology for Adaptable, Reliable Systems program. He received a bachelor's degree in business management from the University of South Carolina, and a master's degree in computer systems management from the Naval Post-Graduate School.

CISE
4516 Henry Street, Suite 205
Pittsburgh, Pa. 15213
Voice: 412/268-4312
Fax: 412-268-6369
E-mail: reb@cise.cmu.edu

Al Krum is Director, Systems Engineering Sector, IMINT, NRO. At the time of the Risk Management Rollout and Installation, he was Program Director, Enhanced Imagery System (EIS), IMINT.

Risk Management Web Sites

www.eas.asu.edu/~riskmgmt This is Arizona State University's (ASU's) software risk management home page. Links include an introduction to software risk management, risk identification questionnaire, and a risk management expert system.

www.infc.ulst.ac.uk/informatics/ise/se/re/serum.html Software Engineering Risk: Understanding & Management (SERUM) site.

www.ida.liu.se/labs/aslab/people/joaka/risk_bib.html Software risk management bibliography with a compilation of software risk management articles by Barry Boehm, R.N. Charette, R. Fairle, et. al.

www.esi.es/Information/Collections/SoftRisk/tools.html Software Risk Management: Tools, including links to risk track, NASA Software Risk Management Database, Software Acquisition Capability Maturity Model®, v. 1.01, and Risk Management Tutorial.

www.spmn.com Software Technology Conference '99 presentations, May 3-6, allows users to download .ppt files on *16 Practices for Improving Software Project Success* by Jane Lochner, and *Software Process Improvement at PMW-163* by Frank Doherty. Also includes Software Program Managers Network quick links.

www.sei.cmu.edu/legacy/risk/kit/metrics.html This focuses on conveying software development risk status without sending upper management into a panic.

www.rollanet.org/~asemmsd/em-handbook/Abstracts/rsk_tool004.html
Abstracts on risk management

www.sea.net.au/project_management/risk_management
International links on:

- Software risk evaluation service and risk management overview from the Software Engineering Institute.
- ASU's software risk management home page.
- Cost of Risk Analysis System by International Security Technology Inc.
- Mesa/Vista Risk Manager, a collaborative web environment that provides the foundation to support a structured risk management process.
- Department of Defense Data Analysis Center for Software site with case studies, resources, training, discussion groups, software tools, and FAQs.

Evaluating COTS Using Function Fit Analysis

This article presents function fit analysis (FFA), a methodology proven to be successful in evaluating commercial off-the-shelf (COTS) products as they relate to meeting customer requirements. The steps of this function point-based process are discussed to show the benefits of providing this information to the decision-making process for the use of COTS as a development solution.

Today many organizations are attempting to reduce software development effort and schedule by purchasing off-the-shelf solutions rather than building them in-house. This strategy can be very cost effective if the COTS solution meets customer requirements. Unfortunately, COTS solutions have often proven to be a great disappointment. This is largely due to poor *fit* in meeting the required functionality. The result is a major COTS enhancement project comparable to a custom-developed solution in terms of overall project schedule and cost. In one such situation, it was found that the COTS solution fit only 2 percent of the requirements. Ninety-eight percent of the solution would have to be delivered as new development and enhancements to the COTS package. Luckily, in this situation the problem was identified early in the process and the proposed COTS solution was rejected.

The situation described above and similar experiences highlight how important it is to conduct a complete and accurate assessment of how a COTS solution fits the requirements and does not rely on vendor claims of high compatibility. This article discusses a technique that has been successful for projects in evaluating the compatibility of COTS products to customer requirements. The technique, called function fit analysis, is based on function point analysis. Function point analysis is the decomposition of an existing or planned system based on the user's perspective of functional requirements. Function points can be used to evaluate various COTS solutions, select the best solution, and determine the degree of enhancement work necessary to meet customer requirements.

Function fit analysis provides the ability to:

- Document functional requirements in terms understandable to users and technicians.
- Identify the functional gap of the COTS products.
- Quantify the effort necessary to enhance the COTS package.
- Provide input into the *make vs. buy* decision making process.

Since function points are a key element to this process, it is important to understand their definition. "Function points are a measure which represents the functional size of application software" [1]. Function points are a unit of measure that represent the work products of software developers. They quantify the deliverables of the software development process. When combined with other data, such as effort and defects, metrics can be developed to aid in planning and managing software projects. Function points were originally developed as a communication tool for defining functional requirements in nontechnical terms. To accomplish this they describe functionality from the end user's perspective of how it supports one's business functions.

To count function points it is necessary to understand the counting rules as well as the user requirements of the project or system being assessed. For that reason, it is important to have knowledgeable participants and supporting documentation available when conducting the count. The function point process as defined by International Function Point User Group follows.

Function Point Analysis Process

There are three types of function point counts:

1) Development project count.
2) Enhancement project count.
3) Application count.

Determining which count to be produced is the first step in the process. The different types of counts and how they are used in the function fit analysis process are described below.

Development project counts are used to size projects with totally new functionality and include all functions being developed. In function fit analysis, the development project count would compare to the *custom-developed* alternative. Enhancement projects are modifications to existing systems and include application functions that are added, changed, or deleted. This is the type of count used when evaluating specific COTS alternatives in the function fit analysis process. It identifies enhancements necessary in the COTS product for it to meet customer needs. The final type of count is an application count. This is the function point count of any installed system. Once a system exists, this is the function point count of all functions provided to the user regardless of how they were delivered (i.e. developed vs. COTS).

To complete a function point count, user recognizable functions are identified and evaluated. From a user's perspective, a computer application assists him in doing his job by providing five basic functions. Two of these capabilities address the data

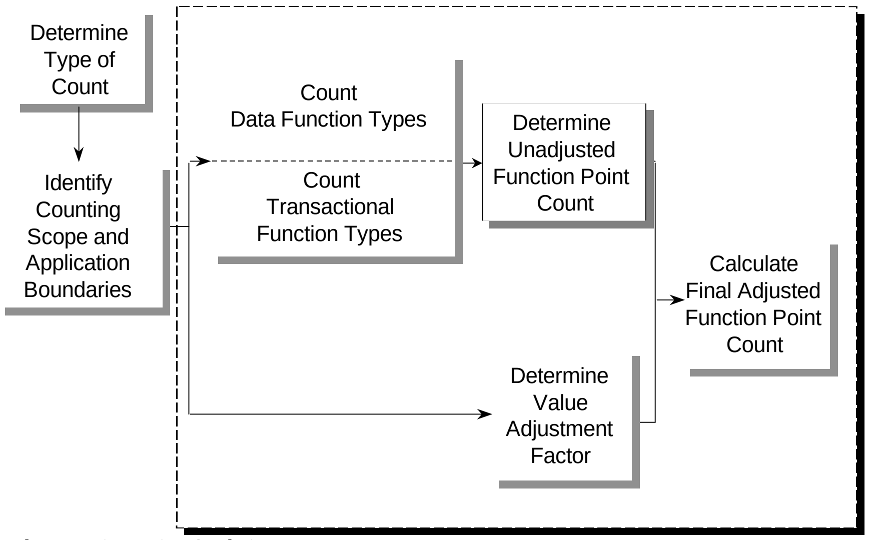

Determine Type of Count
Identify Counting Scope and Application Boundaries
Count Data Function Types
Count Transactional Function Types
Determine Unadjusted Function Point Count
Determine Value Adjustment Factor
Calculate Final Adjusted Function Point Count

Figure 1. *The Function Point Analysis Process*

requirements of the business and are referred to as data functions. Data functions consist of internal logical files and external interface files. Three of these capabilities address the user's need to access and manipulate data stored on the files and are referred to as transactional functions. Transactional functions consist of external inputs, external outputs, and external inquiries.

The Five Components of Function Points
Data Functions
1. Internal Logical Files
2. External Interface Files

Transactional Functions
3. External Inputs
4. External Outputs
5. External Inquiries

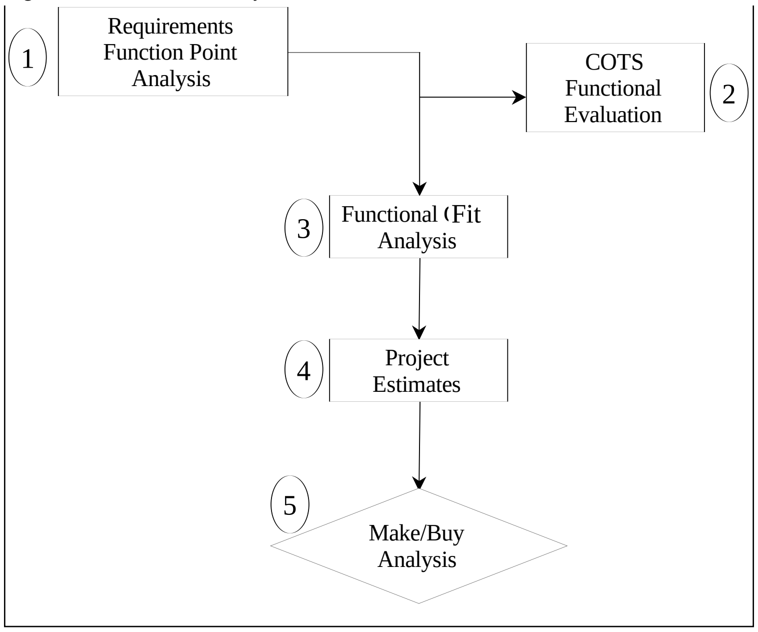
The first data function allows users to utilize data they are responsible for maintaining. For example, a user may be responsible for adding, changing, and deleting employee information on the employee master file. Therefore, the user is responsible for maintaining the file. Logical groupings of data that are maintained by an end user of an application are referred to as internal logical files (ILFs).

The second function of an application provided to an end user is also related to logical groupings of data. In this case, the user is not responsible for maintaining the data, which resides in another application and is maintained by another user. The user of the application being counted requires the other application's data for reference purposes only. For example, a user may require the ability to access security information from the security application. The user does not have the responsibility for updating security information but must reference it to complete his or her job. Groupings of data from another application used only for reference purposes are defined as external interface files (EIFs).

The remaining functions address the user's capability to access the data contained in ILFs and EIFs. This capability includes maintaining, inquiring, and outputting of data, referred to as transactional functions.

The first transactional function allows a user to maintain ILFs through the ability to add, change, and delete the data. For example, a user can add, change, or delete employee information on the employee master file. In this case, the user is utilizing a transaction referred to as an external input. An external input gives the user the capability to maintain data on ILFs through adding, changing, or deleting its contents.

Figure 2. *Function Fit Analysis Process*



The next function gives the user the ability to produce outputs that contain calculations. For example, a user may require a report that contains expense data with derived information. The report is produced using maintained and referenced information. In function point terminology, the resulting report is called an external output.

. . . [Function fit] analysis should only focus on quantifying user requirements, not on potential COTS solutions.

The final capability provided to users through a computer application addresses the requirement to display specific data from files. In this situation there is no manipulation of the data, it is a direct retrieval of information contained on the files. For example, a user can inquire on employee data by inputting a Social Security number and retrieving information. These transactions are referred to as external inquiries.

Two adjustment factors are applied to calculate the function point count. The first adjustment, functional complexity, assigns weights to each functional component based on data elements and processing logic. The second adjustment, the value adjustment factor, evaluates the operational complexity of the application.

Function points are just one piece of information that is used in the function fit analysis process. The function fit analysis process consists of five primary steps.

Function Fit Analysis Process

Step 1: Requirements Function Point Analysis

This step is used to identify and document exactly what functionality is necessary to meet the customer needs. This assessment should be completed early in the development life cycle to provide insight into the scope of the project. It can be completed early because it is based on *what* functions are to be delivered, not *how* they are to be delivered. During this step, a function point count is completed based on documented functional requirements, or by analyzing the functions of an existing system to be replaced. It is important to focus on the *to be* version of the system and to identify all of the functions necessary to meet the users' needs.

The analysis should only focus on quantifying user requirements, not on potential COTS solutions. The result is a development function point count and a detailed listing of the functions necessary to meet the requirements. For example, a course registration application may contain the following user functions:

Add training course information
Modify training course information
Display training course information
Establish training sessions
Add participants to training sessions

Step 2: COTS Functional Evaluation

This involves reviewing the various COTS choices and comparing their functionality to the requirements documented in Step 1. From a function point perspective, this is an enhancement project count, as we are starting with a system and identifying changes necessary to meet the functional requirements. Using the function point count from Step 1 as a guide, each database and panel in a COTS alternative is reviewed to identify functions that:

1. Exist in the COTS with no change required [Unchanged].
2. Exist in the COTS but require enhancements to meet the requirements [Change].
3. Need to be added to the COTS product [Add].
4. Exist in the COTS but are unrelated to the requirements and will not be used [Unused].

The resulting enhancement function point count summarizes how well the COTS product matches the functional requirements. Using the example functions from above, the COTS evaluation function point count contains the following:

User Function	Enhancement Activity
Add training course information	Unchanged
Modify training course information	Change
Display training course information	Change
Establish training sessions	Add
Add participants to training sessions	Add

The COTS enhancement project function point count will include functions that are being added and changed. The first analysis of results should compare the function point size of the COTS enhancement project to the function point size of the custom-developed solution. If the enhancement function point size is not significantly less than the custom-developed solution, then the COTS solution might not be the best *fit*. The second step of the analysis is to evaluate the *unchanged* functions since this represents what portion of the solution *fits* the user requirements. The *unused* functions show how much of the COTS product is unusable and unrelated to the requirements.

Step 3: Functional Fit Analysis

This step is used to apply a percentage to the *fit* between the requirements and the COTS product. By comparing the function point counts from the previous two steps a fit can be determined. One of the first tasks is to define what *fit* means in the minds of all those involved in the assessment. There are different ways of looking at fit depending on the approach used. In function fit analysis, *fit* is defined as the amount of out-of-the-box functionality that can be utilized without any modifications. Using this definition, a comparison of the requirements function point count to the COTS function point count will result in calculating the *fit* percentage of the COTS (FFA fit = unchanged FPs ÷ total FPs from Step 1).

In addition to the *fit* calculation, analysis of the percentage of added and changed functions is also useful. These percentages can be calculated by identifying the *added* and *changed* functions from Step 2, calculating a function point count, and evaluating it against the total function point count from Step 1.

The *added* function point count will show the amount of user requirements that are not supported at all in the COTS product. Obviously, if this is a large number, it may be better to look at a different alternative or consider a custom-developed solution.

Identifying *changed* functions is helpful in evaluating whether to enhance the COTS to meet the business requirements, modify the business process to meet the COTS, or a combination of both. This analysis provides an opportunity to revisit certain functions with the users to see if the requirements are flexible.

Table 1 details added, changed, and unchanged function points for a project.

Identifying and quantifying the *unused* functions in the COTS product can be helpful in a couple of ways. They are used to determine how much of the purchased COTS solution will not be utilized. Also, reviewing these functions with users may help to generate ideas on how to improve the business process in the future.

Knowing the *fit* numbers is one piece of information that is helpful in the make/buy decision. The other piece of information is the estimate to deliver the functions in the various options, which leads us to the next step, project estimates.

Step 4: Project Estimates

The function point data from the previous steps is used to develop the project estimates in Step 4, which uses a top-down approach to estimate effort, staff, and schedule for each alternative under review. The first is for the *custom-developed* alternative. Two components are used to develop an effort estimate: project size, which is the function point count from Step 1 of the function fit analysis process, and an evaluation of the project attributes, factors that, in addition to project size, influence software development productivity. Four major attribute categories are evaluated:

1. Personnel and Management—focuses on knowledge and experience of information systems and end user personnel involved in the project.
2. Process and Methods—focuses on what development and project management methods are used and the extent the methods are followed.
3. Technology and Tools—evaluates the technology being utilized and the effectiveness of the tools available to the developers.
4. Environment and Support—evaluates the development environment in terms of computer resources, administrative resources, and overall working environment.

Once the size and the project attributes are determined, an effort estimate can be developed. This can be done using various industry tools, industry benchmark data from external consultants, or organization-specific historical data. The estima-

Total Project		Added	Changed	Unchanged	Total AFP
	System A	750	350	100	1,200
	System B	800	250	50	1,100
	Total	1,550	600	150	2,300
	Percent of Total Project	67%	26%	7%	100%

Table 1. *Added, Changed, and Unchanged COTS*

tor selects the appropriate productivity rate to use based on the development project function point size and the project attributes. To calculate the effort estimate, the function point count is divided by the productivity rate (project effort = custom developed project size ÷ function points/hour). Schedule and staffing estimates are derived from the project effort incorporating organizational constraints, e.g. staff limitations and preset schedules.

A similar process is followed for each COTS option under review. For this analysis, the project size is based on the enhancement-function point count completed in Step 2 of the function fit analysis process. Project attributes should be reviewed to see if the previously completed assessment applies to the COTS alternatives. Enhancement productivity rates differ from development productivity rates depending on project size, so the effort estimates for the COTS alternatives may not use the same productivity rate (function points/hour) as the custom development option.

Now there is enough measurement data about the functional requirements to feed to the make/buy decision step.

Step 5: Make/Buy Analysis

The function point information and estimate data from Steps 1-4 is used in Step 5, the make/buy analysis. The make/buy analysis is the decision-making process to determine whether to implement a COTS solution or build a custom one. The following are some initial decision points based on information from previous steps:

Buy *As Is* if:

1. The users are prepared to live with the COTS functionality.
2. The users are willing to change their business processes to adapt to the COTS application.
3. The sensitive schedule is an overriding factor. This means delivering the functionality in a certain time frame takes precedence over how the users would like the business process to be. The schedule is the highest priority and it drives the decision.
4. Development and future maintenance funding is limited. If there is not going to be any money available to maintain or enhance a modified or developed system, then it may be best to rely on the COTS provider for enhancement and maintenance support.

Customize COTS if:

1. The cost to customize the COTS product is more viable than building. Development costs as well as ongoing support and operation costs should be included.
2. Minor customization is necessary to meet the user requirements.
3. The schedule is relatively sensitive. If developers can take advantage of existing COTS functionality and modify a minimal number of functions, then the development time frame will be shorter. If a short schedule is a requirement, then business functions should be evaluated to determine if they could be changed to utilize the COTS functionality, therefore limiting the customization.

Custom Develop (no COTS) if:

1. A significant number of requirements are not available. If the user requirements are specific and cannot be changed to use the existing COTS functionality, then building them from scratch is the best option.
2. Initial cost of the COTS, including enhancement and support dollars, is higher than developing and supporting the application in-house.
3. Ongoing upgrades are cost prohibitive. This point considers the cost of implementing future upgrades to the COTS product. If installing the upgrades requires rework of the customizations, then it is possible that significant development effort would be incurred each time an upgrade is installed. If this situation occurs, it would not be cost-effective.

In addition to evaluating the development effort the following factors should be included in the make/buy analysis:

COTS costs
Make sure to include all the costs associated with the COTS product. This would include evaluation costs, initial purchase costs, seat or site license costs, and annual support and operation costs.

Access to package specifics
Discuss with the COTS vendor what technical components of the product will be available to the in-house developers. For example, will they have access to the data model or process model? What *hooks* are available to the developers for adding custom code? Is there a coding standard the COTS provides to ensure custom-built components will have the same look and feel as the core product?

Ownership of customized software
Once the COTS solution has been purchased and customization is completed, there may be a gray line defining who owns what. Make sure you understand the rules up front.

Customization responsibility
Determine at the start who will make the changes to the COTS, and what the associated costs will be. This may also dictate who owns the customized code.

Existing database structure
Some organizations have strict guidelines on naming conventions and structures. It is beneficial to examine the COTS database structure to see if it can easily comply with your organization. If it does not, a decision can still be made to change the organization rules to meet the COTS.

Existing computing architecture
This will be important if the COTS product needs to communicate with other in-house, non-COTS, applications. The compatibility should be examined to determine the amount of work necessary to integrate multiple systems.

Once the above information is gathered and analyzed, the function fit analysis process is complete, and an appropriate make/buy decision can be made. The data and findings from the process should be documented and stored in a repository for reference. This information may be helpful in future analyses of this kind.

Summary

There is a great deal of information that is necessary to make an informed decision as to whether to utilize COTS in development efforts. The function fit analysis process provides an excellent framework for information gathering, evaluation, and decision making. It communicates functional requirements in objective terms, so the COTS can be evaluated from a *fit* perspective and identifies functions to evaluate for possible business process re-engineering efforts.

As function fit analysis provides a size metric for the development options, it enables estimates to be made in terms of effort, staff, and schedule. Without knowing *what* needs to be delivered, it is impossible to determine a good estimate of how long it will take to deliver it. Function fit analysis gives us the *what*.

In today's world of providing things better, faster, and cheaper, it can be difficult to determine the best option. The use of COTS products has been touted as a faster and better method for software development, but that is not always the case. Function fit analysis is a valuable technique to help evaluate purchased/customized solutions and make the best and most appropriate decision for each organization and project.◆

Reference

1. IPFUG Counting Practices Manual v. 4.1.

About the Author

Lori A. Holmes is a Senior Management Consultant with Q/P Management Group, specializing in Total Quality Management, software measurement, and process improvement. Prior to joining Q/P, she spent nine years at First Data Corp. in Omaha, Neb. as Quality Assurance Manager and Applications Programming Manager. Holmes received a bachelor's degree in business administration, business information systems focus, from Illinois State University in 1984, and is a certified function point specialist and a quality analyst.

Q/P Management Group Inc.
15539 Burdette St.
Omaha, Neb. 68116
Voice: 402-493-0228
Fax: 402-493-0506
E-mail: lholmes@QPMG.com

Web Addition

A Ship Cost Agent for Pier and Port Management

Agent-based collaborative decision support is a methodology utilizing a domain specific intelligence system to partner with human decision makers to reach a consensus solution to a complex problem. This paper describes the cost agent and its deployment within a collaborative planning facility management tool designed to support Navy pier and port management, the Collaborative Infrastructure Asessment Tool .

James A. Sena
College of Business, California Polytechnic State University

Find this article on the Internet at http://www.stsc.hill.af.mil/crosstalk/2000/feb/sena.asp

New CMMI Requirements for Risk Management

A new risk management process area is one of the changes proposed in the new Capability Maturity Model Integration® (CMMI) product suite. For some, this is long overdue in the software community; for others, this is a new and potentially difficult requirement.

The purpose of risk management is to identify potential problems before they occur, so that risk-handling activities may be planned and invoked to mitigate adverse impacts on achieving objectives. Risk management is a continuous, forward-looking process, integral to both business and technical management. It encompasses:

- Defining a risk management strategy.
- Identifying and analyzing risks.
- Developing and implementing risk mitigation plans.

In Software CMM v. 1.1, risk management was identified as a key practice in Software Project Planning, Software Project Tracking and Oversight, and Integrated Software Management.

At Maturity Level 2, projects identified, assessed, documented, and tracked software risks. At Maturity Level 3, projects were expected to actively manage the software risks in a more proactive and integrated fashion.

In practice, many current software projects give lip service to risk management in the CMM by listing a vague set of nonspecific risks (e.g., inability to meet schedule). Projects fail to identify or quantify project-specific risks or mitigation plans, or to continuously manage them throughout the project life cycle. Often, practitioners think that risk management is a systems engineering function, outside software's responsibility.

In creating the CMMI-SW/SE v. 0.2 maturity model, risk management was elevated to a process area, with a full set of required goals and expected practices that apply to software. This structure follows the System Engineering maturity models, where risk management has always been handled at the process area level.

In addition to specific project activities, risk management must be institutionalized through written organizational policies, plans, allocated resources, assigned responsibilities, training, configuration management of work products, quality audits, and management reviews.

Some organizations have already reaped the benefits of a mature, proactive, software risk management process. Resources include the SEI's Software Risk Taxonomy and Team Risk Management process, the Software Project Manager Network's Risk Radar tool® , and numerous books and training courses.

Contributed by Rick Hefner, Ph.D, TRW
Co-Chairman
CMMI Assessment Methodology Team
rick.hefner@trw.com

Give us your information–get a free subscription

Fill out and mail this form to us for a free subscription to CROSSTALK.
7278 FOURTH STREET, HILL AFB, UTAH 84056 ATTN: HEATHER WINWARD

Or save a stamp and subscribe online at http://www.stsc.hill.af.mil/request.asp

FULL NAME:______________________________

RANK OR GRADE:______________________________

POSITION OR TITLE:______________________________

ORGANIZATION OR COMPANY:______________________________

ADDRESS:______________________________

____

BASE OR CITY:______________ STATE:__________ ZIP:__________

VOICE: COMMERCIAL______________ DSN:______________

FAX: COMMERCIAL______________ DSN:______________

E-MAIL: ______________@______________

1999 BACK ISSUES AVAILABLE (CHECK THE BOX OF THE MONTH(S) DESIRED.)

JANUARY ☐ FEBRUARY ☐ MARCH ☐ APRIL ☐ MAY ☐ JUNE ☐

JULY ☐ AUGUST ☐ SEPTEMBER ☐ OCTOBER ☐ NOVEMBER ☐ DECEMBER ☐

Get some information from the STSC

The Software Technology Support Center (STSC) offers a large selection of software engineering documents and tool and technology reports online. The STSC also specializes in fee-for-service consulting services with expertise in areas such as

- *Measurement*
- *Project Management*
- *Re-engineering*
- *Process Improvement*
- *Acquisition/SA-CMM*
- *Peer Reviews and Defect Prevention*
- *CMM Assessments*
- *Documentation and Standards*
- *Configuration Management*
- *Object-oriented Development*
- *Personal Software Process*
- *Software Quality Assurance*

Contact Rudy Alder at 801-777-2750 DSN 777-2250 or e-mail rudy.alder@hill.af.mil for more information

For those of you who wrote to remind us that our references to the end of the millennium in the December issue were premature if we believe the math instead of the marketing, thank you, and please visit these sites: www.enigmar.com/2001 www.astronet.com/features&fun/millennium.html www.georgian.net/rally/madness.html

Thank Goodness We're Not Dumb Animals

BACKTALK

Gosh, monkeys sure are dumb. Here is a case in point:

Five monkeys were in a room that contained a table in one corner, and a banana hanging from a string in the middle of the room. The monkeys figured out that if they dragged the table to the middle of the room, they could climb up and grab the banana. So they did. As one of the monkeys quickly hopped up and reached for the banana, hidden compartments in the walls suddenly opened, releasing high-pressure cold water that knocked the monkey off the table and drenched the other four monkeys.

They quickly learned that whenever one of them climbed on the table, all of them were soaked with cold water. They realized climbing on top of the table was a bad idea. Unbeknownst to the monkeys, the high-pressure cold water hoses were disconnected and removed.

The next week, one of the five monkeys was removed from the room and replaced by a new monkey. The new monkey saw the table and the banana dangling from the ceiling. Realizing that the banana was there for the taking, the monkey headed for the table. But fearful of being drenched by the high-pressure cold water, the other four monkeys pounced on the newcomer and beat the tar out of him. Every time the new monkey got near the table, the others beat him up. Soon the new monkey no longer went near the table.

By the third week, another of the original five monkeys was replaced by a new monkey. And like the monkey the week before, the newest member of the group tried to get near the table to move it over to the banana. Once again, the others beat up the newest member of the group. Even the first new monkey joined in.

Each successive week, one more of the original monkeys was replaced. The same thing happened every time; when the newest monkey attempted to get near the table, the others joined in to discourage him.

By the sixth week, not a single monkey was left from the original group. Not one remained that had been squirted with cold water. But when the newest monkey headed toward the table and tried to reach the banana, the other four monkeys "trained" him by beating the tar out of him.

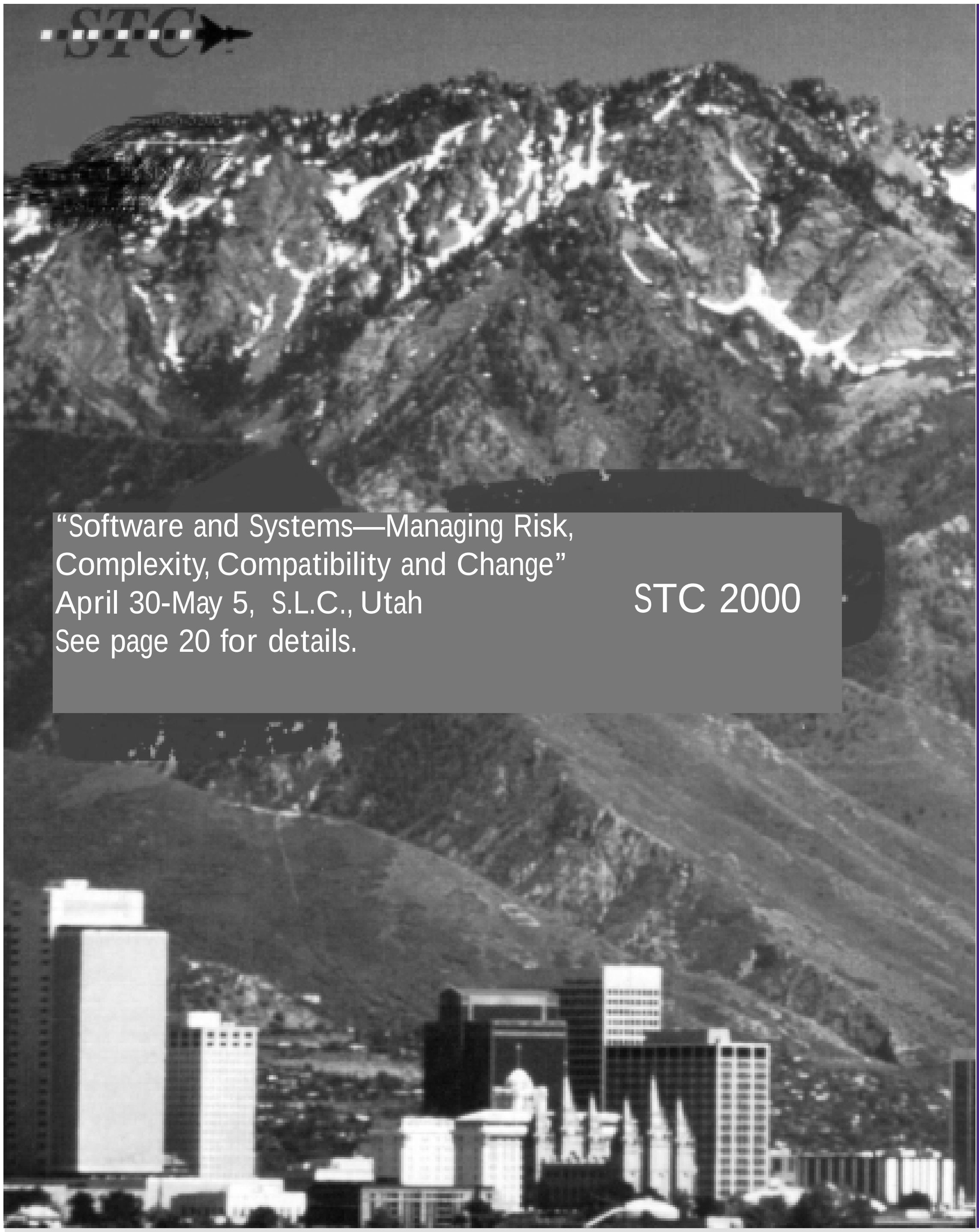
If you could ask each monkey why it was beating up the new monkey, each probably would say, "I don't know, that is just the way we do things around here."

Pitiful, isn't it? Imagine a bunch of dumb primates that not only refuse to try new techniques, but attack, ridicule, or otherwise discourage other primates who try new things. Their only possible defense is that when they tried new techniques, somebody attacked, ridiculed, or otherwise discouraged them. So they continue doing things the old-fashioned way.

Gosh, monkeys sure are dumb. Imagine being unable to take a calculated risk and break an old behavior pattern to try new things!

—Dave Cook, C. S. Draper Laboratory Inc.

Got an idea for *BACKTALK*? Send an e-mail to backtalk@stsc1.hill.af.mil



“Software and Systems—Managing Risk, Complexity, Compatibility and Change”
April 30-May 5, S.L.C., Utah
See page 20 for details.

STC 2000

Published by the Software Technology Support Center

S T S C

Sponsored by the Computer Resources Support Improvement Program (CRSIP)

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

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