

Software Quality Matters

Newsletter of The Software Quality Institute at The University of Texas at Austin in Cooperation with the Austin Software Council and the Austin Software Process Improvement Network

Volume 6, Number 2



Summer 1998

Two-Party Contracts—Part I

Looking at Legal Implications of Unsuccessful Software Projects

by Herb Krasner

(Disclaimer. This article should not be construed as providing any legal advice or a legal opinion, and it does not purport to give a summary of the law or a comprehensive analysis of the legal issues raised therein. Do not act or rely upon legal information or views in this article without consulting an attorney.)

Part I of this article examines the nature of legal actions in two-party software development situations.

One of the unfortunate aspects of today's complex software systems development projects is the possibility of litigation. Legal issues for software systems include intellectual property rights, licensing, tax issues, contract award protests, fraud, and breach of contract. The software issues in these areas are not as well understood as in more traditional products/systems.

As software professionals, we tend to think these matters are for attorney attention; a company's success in litigation, however, depends directly on what is done during the development process.

I have found many management and technical software professionals who could care less about the legal world. If this is your view, it's a dangerous one, for you are putting your company at risk. Defending against a lawsuit is serious and expensive business. This article focuses on two-party contract development situations.

The Unsuccessful Project

An unsuccessful software project is one that does not fully achieve its stated objectives. These objectives involve a software system that works as promised, is delivered on the promised date, is within established cost limitations, and that satisfies various other constraints defined in a contract.

Contracts usually establish phase-oriented intermediate milestones on which progress toward end objectives can be determined. Early indications of a possible failure to perform against objectives usually shows up as schedule slips at early milestones as the organization struggles to produce interim work products (e.g., a software development plan, a functional specification, etc.).

Following are possible negative project outcomes.

- The system is delivered on time but in an unacceptable condition.
- The schedule is allowed to slip significantly so that an acceptable system may be delivered.

- More money or more time is requested because of significant changes in system scope.
- The project is determined to be irrevocably off track and is prematurely canceled.

A negative outcome can have significant monetary implications. Lawyers are concerned about who is responsible for the lack of success and who owes whom as a result. It's not just the sunk development expenditures that are at issue but also the cost implications for the customer who does not have the promised system in place.

Reasons for Ending up in Court

There are many reasons a company responsible for an unsuccessful software project may end up in court.

1. Committing a criminal act—software would be the means of committing a crime
2. Liability—defective software led to a failure that caused personal injury, property damage, or large monetary loss
3. Negligence—the same as liability but also including the additional concern about whether or not a reasonable process was used to develop and test the software
4. Fraud—knowingly making false statements in order to coerce the outcome of an economic decision

Legal Implications (continued on p. 2)

What's Inside

Architecture Conference	3
SQI Book Series	4
Multimedia Conference	4
PSP Course Begins	5
A-SPIN Program Series	6
Conducting FTRs	9
Paulk's Q&As	11

Intensive Personal Software Process Training Course to Begin September 11th

Legal Implications (continued from p. 1)

5. Negligent misrepresentation—fraud without knowing that statements were false when made, assuming that due diligence in fact finding was done
6. Deceptive and/or unfair trade practices—practices that exhibit a pattern of misconduct (e.g., false advertising, making vaporware delivery date announcements, etc.)
7. Unfair competition—a competitor sues based on a pattern of illegal activities
8. Malpractice—negligence judged against accepted professional standards of workmanship (e.g., software developer certification standards)

Contracts get into trouble quickly when the customer or the developer uses a different set of basic working assumptions about the relationship.

9. Breach of contract—a general set of contract laws apply for custom development and service contracts; these mostly depend on the type and specific terms of the contract; for consumer products, licensing agreements and UCC Article 2 (Law of Sales) apply
10. Grossly negligent provision of services—whether or not the service provider made reasonable efforts to provide the service
11. Violation of government regulatory agency requirements (e.g., FTC, SEC, FDA)

Opposing Views

Litigation usually occurs when the level of trust between the customer and the developer disintegrates as the result of involvement in an unsuccessful software project, and the motivation to blame and punish then takes over. Regardless of the causes, lawsuits can result in plenty of bad feelings, misrepresentation of perceived facts, misunderstandings, and miscommunication.

The customer may claim that the developer:

- Failed to perform relative to contractual obligations (e.g., did not deliver various interim/final products and documents on schedule),
- Mismanaged the project (e.g., did not staff with qualified people in a timely manner),
- Misrepresented its abilities to perform the job successfully, or
- Underestimated the project intentionally to win a competitive bid.

The developer may claim that the customer:

- Tasked him to do an impossible job,
- Provided him with an incorrect or incomplete set of requirements,
- Misscoped the project and purposefully underdescribed critical areas, or
- Changed the requirements uncontrollably *in situ*, thus leading to unstable development conditions.

Given these two opposing views, a lawsuit attempts to determine which view is essentially correct.

With some thought we can see that these two opposing viewpoints are based on a different set of assumptions about the relationship between customer and developer.

Following are some of the assumptions that can critically affect customer/developer relationships. These assumptions are too often unstated and untested.

1. The customer should know how to define his problems and his needs for the system to be developed.
2. The customer doesn't know much about how to design a software system solution for his problems and needs.
3. The developer doesn't fully understand the customer's problems/needs or the context in which the solution system needs to fit.
4. The developer is knowledgeable in the solution technologies that must be used to create the system.
5. Both parties share a common understanding of the areas of the unknowns in the problem/solution space and are willing to take those risks.

The number of us who forget about these basic assumptions when we enter into a contractual agreement always amazes me. Contracts get into trouble quickly when the customer or developer uses a different set of basic working assumptions about the relationship and about who should and does know what.

Common Issues of Unsuccessful Projects

The common issues associated with most unsuccessful projects arise essentially where mismanagement and a lack of understanding of the nuances/dynamics of complex system projects intersect. At least part of the cause for this is, I believe, because good software management and good software engineering skills are scarce, and the combination of the two is even more rare.

Root causes of unsuccessful projects are difficult to determine because of different perceptions of what happened. We often hear of the following complications and/or symptoms.

- Ambitious attempts to do too much with too little
- A politically defined grab-bag of unrelated requirements
- An unprecedented software system for an ill-defined problem
- Poor customer developer relations
- A flawed contractor selection process
- Lack of agreed-to expectations for project success and the level of risk to be taken by each party
- A lack of specific acceptance criteria
- A lack of understanding of what capabilities the users really need
- Quantitatively uncontrolled project scope
- Complex and volatile technical requirements
- Unreasonably restrictive imposed constraints
- On the leading edge technologically
- A poor software definition, development, and delivery process

Preparing for Litigation on the Developer's Side

Since this article is for software developers, let's assume that we are trying to build a defensible case for the development organization. What we need are facts, and it would be nice if they already existed.

Step one in preparing for litigation is to initiate fact finding. Evidence is gathered and then built into plausible explanations (hypotheses) that can support or refute the various claims and counterclaims. It is the strength of this evidence and the ability to explain it to a jury or arbitrator that influences the litigation result one way or the other.

Legal Implications (continued on p. 3)

Invitation-Only Conference On Software Architecture Slated for February 1999



Organizers of the First Working IFIP Conference on Software Architecture (WICSA1) have issued a call for papers. Conference attendance is limited to 100 and is by invitation on the basis of a submitted paper. The deadline for submissions is July 15, 1998.

Sponsored by IFIP TC-2 and organized by the Software Engineering Institute, the meeting will be held Feb. 22-24, 1999, in San Antonio.

WICSA1 will provide a focused and dedicated forum for the international software architecture community to unify and coordinate its effort to advance the state of practice and research. An important goal of this working conference is to facilitate information exchange between practicing software architects and software architecture researchers. This conference will serve as a kickoff event for a new IFIP TC-2 Working Group on Software Architecture and will shape the focus and tasks of the working group for the initial period.

Technical, Experience, and Position Papers Sought

The conference is soliciting papers on all aspects of software architecture describing results of the research, research in progress, case studies, experience reports,

and position statements. In particular we encourage submissions from industry that present or analyze experiences of software architecture for complex systems and expose problems in the design, implementation, or evolution of software architectures.

Following are specific topics of interest.

- Theory
 - Architectural models
 - Architectural descriptions
 - Rationale (relating problems to solutions)
 - Analysis, evaluation, and validation
- Processes
 - Architecture-based business or economic models
 - Architecture-based development
 - Architecture extraction and recovery
 - Product line (reuse-based processes)
- Technology support for
 - Creation/evolution
 - Analysis, evaluation, and validation
 - Platforms, product lines, cross-product sharing
 - Configuration, customization, generation
 - Dynamic architectures

- Example architectures
 - Domain-specific reference architectures
 - Styles, patterns, and codifications
 - Product line architectures
 - Cooperating architectures
- Training
 - Skills and traits for good architects
 - Education requirements
 - Standard architectural examples

Conference Speakers and Working Sessions

Keynote speakers will discuss where we are, where we should be, and where the hard problems and the strong leverage points are.

Conference working sessions will address similar issues.

- What are the most difficult tasks performed by practicing software architects and what is available from research to help solve them?
- Where are the gaps between business needs and research results, and what can be done to bridge these gaps?
- What important problems are being addressed in research and why are they (or why are they not) relevant to practicing architects?

Submission Guidelines

Deadline: July 15, 1998

See SQI website for details:

www.utexas.edu/coe/sqi

“Upcoming Software Conferences—Software Architecture”

Legal Implications (continued from p. 2)

Unfortunately, in too many cases, the development organization does not have the evidence needed to support its position. This lack of information may correlate to how the developer got into trouble in the first place; that is, had the information been available during the project the developer could have used it to prevent the problem from growing to the point of litigation.

What kinds of evidence are we talking about? Metrics, defined processes, and documented expectations.

Metrics. The most valuable pieces of evidence are measurements and chronological trends. Basic software metrics should measure and track (1) technical size/scope, (2) development effort, (3) schedule progress, (4) quality, and (5) levels of project uncertainty and risk. Too many organizations do not measure even these basic items and, thus, have no indicators of trouble until it's too late.

Processes. Defined and reasonable management/development processes and practices are evidence of professionalism. These should be part of a well-integrated strategy and project plan for achieving success according to the stated objectives. A CMM Level 4 process and related process quality metrics would be very convincing in

fighting off a negligence lawsuit. Organizations operating without any kind of defined process are at a serious disadvantage.

Divergent Expectations and Assumptions. The project's objectives and criteria for success should be defined, mutually understood, and agreed to by customer and developer. When the customer's expectations for success begin to deviate from the developer's expectations for any reason, they should be documented (as issues) and action should be taken by management. An organization that does not continually test its understanding against the customer's understanding is at serious risk.

Part II

In the second and final part of this article, which will appear in the next issue of *Software Quality Matters*, the author will suggest actions software professionals can take to successfully ward off litigation. These professionals include software and database engineers, software project managers, business unit and senior management, and software quality engineers.

Copyright © 1998 by Herb Krasner. All rights reserved.

Multimedia Computing Conference Slated for June 29th in Austin

Several hundred representatives from multimedia organizations worldwide are scheduled to gather on June 29 through July 1 at the Renaissance Austin Hotel in Austin for the IEEE International Conference on Multimedia Computing and Systems. The event is expected to draw professionals from 14 countries in North America, Europe, Asia and Australia.

The conference will bring together researchers, developers, and practitioners from academia and industry working in all facets of the multimedia industry, according to local chairman Gary Cobb of Dell Computers.

This is an exciting event for us to host in Austin, Cobb says. Previous conferences have been held in Boston, Canada, and Japan with the next one set for Florence, Italy.

Harrick Vin of UT's Department of Computer Sciences, program chair, has lined up an impressive range of topics for tutorials, technical papers and workshops.

Technical topics will prevail along with other presentations of a more general appeal such as the virtual recovery of excavated archaeological finds at China's Museum of Terra-Cotta Warriors and Horses.

Other topics include web page caricatures, the construction of an interactive movie system for multi-person participation, and the design of MiMaze, a multi-player game on the Internet plus dozens of technical topics.

Keynote for the conference will be Jim Foley, chairman and CEO of Mitsubishi Electric Information Technology Center in Boston, who will look at "User Interfaces of the Future." In his talk, Dr. Foley will touch on directions of user interfaces, changes in paradigm of how certain tasks are performed, and the vision of closely coupled human inputs and outputs.

On Monday evening, June 29, a special event produced by Austin's premier game company Electronic Arts' Origin Systems, will feature a presentation by Richard Garriott, CEO of Origin Systems, on how games are developed. As part of the talk, Garriott will also discuss new technologies required by game companies of the future.

The Origin presentation, which is open to the general public starting at 7 p.m. in the ballroom of the Renaissance Austin Hotel, will include opportunities for participants to play Ultima Online, Origin's internet-based game.

First Master's Class in Executive Software Engineering Graduates

SQI heartily congratulates the students in UT's first Option III master's degree program in Executive Software Engineering who graduated in December.

The master's degree program is designed for software professionals who work full time. Classes in the program meet once a month, all day Friday and all day Saturday over a two-year period. The program is offered by UT's Department of Electrical and Computer Engineering and administered by SQI staff.

New students are admitted each year in January. The priority deadline for applications is August 1 preceding the requested January start date. If you would like additional information about this program, please call (512) 475-6779 or send an e-mail request to elms@sqi.utexas.edu

Graduates of the Executive Software Engineering master's degree program with an emphasis in Software Engineering represent a broad range of software organizations in the Austin area.

Willie Adams (Dell)
Deepika Arora (National Instruments)
Amy Bearfield (Motorola)
Jeff Coleman (M/D Totco)
Steve Dobbstein (IBM)
David Gaertner (IBM)
Steve Houser (Applied Research Labs)
Judy Lanoux (Evolutionary Technology)
Nancy Nixon (Wayne-Dresser)
Jim Poag (Applied Research Labs)

Ashley Rosilier (IBM)
Leigh Williamson (IBM)
Mabel Willy (Applied Research Labs)
Glen Wilson (Motorola)
Russell Wilson (ALIVE)

Apply Today for Class That Begins January 1999

Are you interested in applying for the Executive Software Engineering master's degree program that starts in January 1999? Now's the time to start the application process.

Classes will meet all day Friday and all day Saturday, once a month, for two years at the convenience of students who work full-time. Summer classes are independent projects.

Applications are reviewed on a first-come, first-served basis. So apply early to meet the August deadline.

These are the procedures:

1. Contact Carolyn Elmshaeuser for an application packet. (512) 475-6779 or send email to elms@sqi.utexas.edu
2. Arrange to take your Graduate Record Exam (GRE) electronically at Sylvan Technology Center (512) 441-1978.
3. Request prior transcripts from universities attended.
4. Request three letters of recommendation, including a recommendation from your manager.
5. Submit your application as soon as possible.

New Book Series from SQI And Macmillan Technical Publishing



SQI and Macmillan Technical Publishing have formed an exciting partnership to publish an authoritative series of books on key topics for software professionals.

The books will detail real-life problems and offer strategies for software managers, developers, and business leaders. Their authors will be highly skilled, experienced

There will be exhibits of commercial products along with demonstrations of non-commercial works by Austin's companies and academic research groups.

For full details and registration information, see: www.utexas.edu/coe/sqi

practitioners who understand critical issues facing today's rapidly changing software industry.

The SQI Series will focus on the following three areas.

1. Software Development Practices and Technologies
2. Management of Software Organizations
3. Integration of Quality Software into Businesses

An editorial advisory board established by SQI is responsible for keeping the vision for the series focused and for ensuring that each project makes a contribution

Book (continued on p. 5)

Intensive Personal Software Process Training Course to Begin Sept. 11



The long-awaited intensive seminar on Personal Software Process (PSP) will begin September 11. SEI-trained Jim Terrel will be the course instructor.

Sponsored by SQI and meeting on Fridays through November 20, the 80-hour course will offer software professionals a means for evaluating their performance and steadily improving their practices.

Developed by Watts Humphrey for Carnegie Mellon's Software Engineering Institute (SEI), PSP training shows software engineers how to (1) define their personal processes; (2) estimate, plan, and track work; and (3) produce defect-free software products. Developers trained in PSP have been shown to achieve a 5- to 10-fold increase in software productivity and quality.

Meeting Dates

The class will meet in the Commons Building at UT's Pickle Research Campus from 8:30 a.m. to 5:00 p.m. Lunch will be provided.

Book (continued from p. 4)

to the field. Al Dale, executive director of the Software Quality Institute, who is responsible for initiating the book series, serves as Chairman of the Editorial Board. Other members are:

- Les Belady, retired chairman and CEO, Mitsubishi Electronic Information Technology Center
- Peter Freeman, dean of the College of Computing, Georgia Institute of Technology
- Herb Krasner, president, Krasner Consulting; consultant for software quality processes
- John Musa, consultant for software reliability
- Shari Lawrence Pfleeger, consultant for software metrics
- Anthony Wasserman, president, Software Methods & Tools

Looking for Authors

Would you be interested in writing a book for the series? SQI is looking for authors who want to share their experience with the software community.

The ideal author for a SQI Series book has several years of practical experience, technical competence, and the ability to

Meeting dates are on Fridays: Sept. 11, Sept. 18, Sept. 25, Oct. 2, Oct. 9, (one week off) Oct. 23, Oct. 30, Nov. 6, Nov. 13, and Nov. 20.

Course Requirements

Attendance is mandatory. Each student should bring a laptop computer with Windows environment to class. SQI will provide Excel Spreadsheet and Microsoft Office 97 for student use during the seminar. Students will use the same language they normally use in the workplace.

SQI also will provide students with the textbook *A Discipline for Software Engineering* by Watts Humphrey.

Students will be assigned advanced readings before the class begins.

Homework

PSP students will have weekly programming assignments that will require approximately eight hours to complete. Homework will be due before 5:00 p.m. on the Monday following each class.

Certification

Certificates will be awarded to students successfully completing the course.

Registration Fees

The registration fee for nongovernment employees is \$3,000 per student. For government employees, the fee is \$2,700 per student. A 20 percent discount is offered when five or more students from the same organization register as a group.

Reservation Information

To reserve seats in the class, contact—
Marilyn Robertson
Phone: (512) 475-8649
E-mail: marilyn@sqi.utexas.edu

How About These PSP Short Courses?

SQI is considering offering the following PSP training in the future.

- PSP for Senior & Executive Managers—½-day seminar
- PSP for Project & Product-Line Managers—⅓-day seminar

If your organization is interested in either of these programs, please let us know. Contact marilyn@sqi.utexas.edu

Straight from the Horse's Mouth—PSP Instructor Terrel Speaks Out

by Cinda Cyrus

If you've been a programmer for any length of time, you've probably had it up to *here* with talk of process improvement for the team and the organization and the company.

communicate an idea or development strategy clearly.

We are also looking for practitioners with suggestions for book titles and an interest in reviewing manuscripts and book proposals, according to Dale.

Send Your Ideas

SQI wants to know what you recommend for the series, so send us your ideas! Contact either Joanne Click, director, Software Quality Institute at click@sqi.utexas.edu or Ann Trump Daniel, executive editor, Macmillan Technical Publishing at adanield@texas.net

But how about process improvement for the person who does the Real Work: the programmer? You know—the solitary individual, sitting in the glow of the monitor, clicking away for unsung hours as the deadline looms.

After all, it's here, with the individual, where quality starts—or, God forbid, ends. And it's this person in this place that Watts Humphrey has targeted for quality improvement.

With his Personal Software Process (PSP) training, Humphrey focuses on the individual, offering a way for the programmer to improve not only his or her planning skills but also the actual quality of the code produced. It goes without saying that better contributions by individuals improve the team *and* the bottom line.

Straight (continued on p. 6)

Straight (continued from p. 5)

This September, SQI offers its first intensive PSP training course, and Jim Terrel, the PSP instructor, is a believer, as the following interview shows.

Question: *What is PSP?*

Terrel answers: PSP is the mnemonic for Watts Humphrey's Personal Software Process, and he describes it thoroughly in his book *A Discipline for Software Engineering*. In a nutshell, PSP is really the application of very well-known scientific/engineering principles to your own work.

Question: *What principles?*

Terrel answers: Basically data gathering and data analysis so that you may understand the way you work, how long it takes you to perform certain activities, what sorts of errors you tend to make, and so forth.

Question: *Why do you need all these data? What are the goals of PSP?*

Terrel answers: There are actually two major thrusts to PSP: improving planning and improving quality. The underlying notion is that the quality of a software system is governed by the quality of its worst components—the old weakest-link-in-the-chain idea.

It's like this. The quality of each component is governed by the number of defects injected and removed by the individual who constructed the component, and the rates of defect injection and removal are affected by the individual's knowledge, discipline, and commitment. If the person doesn't know *how* to do something—that is, lacks the knowledge—then he or she needs training if this individual is to create high-quality products.

This really isn't a PSP issue except in the sense that a person may have the computer science knowledge but lack the programming skills to produce high-quality products. By creating a stable personal software development process that can be analyzed, a person can better apply his knowledge.

If the individual lacks discipline, then he or she tends to make lots of sloppy errors. The person knows better but hasn't taken the time or applied the effort to find the defects early in his or her own process. "The compiler or unit test will find'em."

Likewise, if the person has accepted an unreasonable commitment due to schedule pressure or just poor scoping and estimation of the time and effort involved to perform a task, the individual is also likely to produce low-quality products in an attempt to meet the commitment made to the manager.

Thus, by standardizing the activities I perform and collecting data (time, defect types, etc.), I am creating a repeatable personal

A-SPIN 1998

Program Series Addressing Human Aspects of Process Improvement Continues

by Geree Streun



The 1998 President and Board of the Austin Software Process Improvement Network (A-SPIN) established a program this year focusing on the human aspects of process improvement. The popular and well-attended programs, which began in January, will continue through November.

Process improvement cannot succeed in an organization unless the people aspect is addressed by the person instigating change, as well as the people accepting and adopting change. For instance, as Celeste Armstrong pointed out, a lot of us have found it more comfortable to continue driving a car leaking oil. All we have to do is carry a case of oil around with us and add another can every 100 miles or so. This is so much easier than changing: getting a new car and having to learn all those bothersome new processes that driving a new car involves!

Let's set the stage for the Process Warrior. All too often, those of us involved in process improvement activities feel like victims, bruised and bloody. What if we change this view of ourselves? What if, instead, we strategically plan our effort? What if we address our battles with specific tools while we engage in this war called process improvement? Then we can see ourselves not as battered victims but as Process Warriors!

The Program through May

At the January A-SPIN meeting, the Process Warrior concept was introduced, and

process that I can analyze and so improve my planning and quality.

Question: *How is the PSP course structured?*

Terrel answers: PSP training is structured around the dual planning and quality goals, with planning emphasized in the first half of the course and quality in the second half.

Each student writes 10 programs during the course. The first program is used to baseline the individual's current process. Then the student is incrementally introduced to new process steps to capture data required for improvement.

the first weapon was presented, "The Five Rules of Communication."

In February, we discovered another weapon via the "Game of Five Questions."

In March, we refined our skills with "The Art of Questions."

In April, we learned about the needed synergy between the P-CMM and the Software-CMM. Real process improvement requires both.

In May, "The Behavior of Change" covered how some of these weapons (skills) have been applied.

The Program from June through November

At the June 18 A-SPIN meeting, a skit will address what really happens on some process improvement quests. Then the SPIN group will build individual workbooks on points for improvement and strategic applications of our weapons (skills).

On August 20, we'll be emerged in both risk and opportunity and learn how to keep them in harmony.

On September 17, "The Five Reasons People Do Not Do as We Expect" will give us our next Process Warrior tool.

"The Five Responsibilities of Leadership", the presentation on October 15, will provide us with yet another tool for our Process Warrior arsenal.

On November 19, the Process Warrior series ends as we are given our last weapon with a presentation titled "Reinforcing Desired Behavior".

Several in-depth analyses and reports concerning the student's personal data are also required. These tie the data together and show the person where and how to improve his or her process.

The course is really very much a hands-on, learn-by-doing experience.

Question: *With all of this data gathering, doesn't your productivity really decline?*

Terrel answers: That's a reasonable expectation, and I certainly had that concern when I entered the course, even though I'd

Straight (continued on p. 7)

Taking No Prisoners at A-SPIN

by Paul Ehrler

It was take no prisoners at January's meeting of the Austin Software Process Improvement Network (A-SPIN), with Stu Penny's spirited presentation of "The Process Warrior."

Eschewing the usual slide-show format, Penny, ex-fighter pilot and cofounder of the Gunpai Corporation, asserted that in today's dog-eat-dog world, we need to emulate medieval Japanese warlords. "We're all in the battle business, and we have to consider any means of achieving our goals," Penny said. Despite the bellicose sounding terminology, Penny does not advocate augmenting our toolsets with Ginsu knives. The key warrior trait he discussed is confidence: a soldier who is not absolutely certain that he can win is at a serious disadvantage and might not survive to fight the next battle.

Different Meanings for Simple Words

Handing out a worksheet, Penny had us record first impressions, Rorschach-style. For example, we were asked to associate pictures and colors with common words. Comparing answers, he showed that even such simple words can have vastly different meanings to different individuals. In business, as on the battlefield, the difference between thinking we know and really knowing can make all the difference in the world, he said.

Straight (continued from p. 6)

seen Watts's graphs that show productivity through unit testing does not really change.

In my own data, with the exception of some drops in productivity when new processes were introduced (i.e., the learning curve), my productivity really didn't change, but the quality of my code increased dramatically (X% fewer defects in compile, Y% fewer defects in unit test).

At first this seems counterintuitive. We've all seen (and generally ignored) industry data showing the relative expense of finding and fixing defects early vs. late in the life cycle. In PSP you discover the same effects, but with your own data it's compelling. And basically the time you save from avoiding compile and test-defect costs more than covers the data-gathering costs. In addition, you've now got data to support effective and realistic planning.

Question: *So, if productivity doesn't improve dramatically, why bother with all this training and paperwork?*

Much of the evening centered around Penny's basic tenet that our common sense notion of what it means to communicate is too broad to be useful. His own animated presentation style exemplifies his definition of the concept. Without constant give and take between presenter and listener, there is no guarantee that the information being exchanged is of value.

The importance of making sure we're on the same wavelength is hardly controversial; it might even be considered self-evident. What Penny adds is that by rethinking the very concept of what it means to communicate, such problems can be managed or even eliminated.

Are All Visual Thinkers?

A few of Penny's assertions seemed a bit over the top. For example, he does not regard classroom lectures or e-mail as legitimate forms of communication simply because they lack the immediate-feedback ingredient. Later on he claimed that everyone, essentially without exception, is a visual thinker. (Besides flying in the face

of a considerable body of research, what does that suggest about blind persons?) Maybe such license can be justified, though. His definitions and generalizations simply bring the underlying theme into sharper focus.

A couple of Penny's views and methods are a little harder to explain, however. His emphasis on immediate response over reasoned deliberation may not be the most effective way to achieve true consensus where none previously exists. While his Socratic juggernaut approach may produce an adrenaline rush, I have some doubts about how effective it is in changing the minds of those who don't already agree with his thesis.

A Success in the Persuasion Business

On the other hand, Penny is a successful player in what could be called the persuasion business, and it would be presumptuous to question the effectiveness of his methods. To be honest, I suspect that I may be a little biased in my assessment. Maybe I'm reading too much into personality types, but it has occurred to me that my Myers-Briggs classification might explain how the delivery could interfere with the message.

Prisoners (continued on p. 8)

Process Warriors, arise!

If you haven't yet discovered this fine program, it's not too late to add some new weapons to your process improvement arsenal. Join your fellow warriors at A-SPIN on the third Thursday evening of each month except July and December. Time: 6:30 - 9:00 p.m., Commons Building at UT's Pickle Research Campus.

Terrel answers: PSP only measures productivity through unit test. Downstream (system test) productivity has been dramatically higher in several real-world industrial settings. In addition, maintenance costs have been extremely low with commercial products produced by PSP-trained teams.

Question: *Did you enjoy PSP training?*

Terrel answers: I would say PSP can be very rewarding, but human nature (or at least mine) does not gravitate towards structure and discipline. Entropy is more likely. It is not *fun* to discover and record every foible. It is rewarding, however, to discover that given a good process, even us old dogs can learn new tricks and make dramatic improvement.

Question: *Isn't it embarrassing to reveal your personal limitations?*

Terrel answers: Sure! But you are actually only revealing them to yourself and the instructor. All the data discussed in the class is aggregate data from the class. The same is true in industrial settings where PSP is being used. You can't expect people to record accurate data if they are punished for doing so! Not even Ratbert in the Dilbert cartoons could pull that off.

Question: *Is PSP effective?*

Terrel answers: It certainly is. SEI has a new technical report (Dec. 97) showing the effectiveness, and I'd recommend that you check SEI's web site for statistics. Consider this. In the data I've seen from some organizations, the *worst* programmer before PSP training was transformed into as good a programmer as the organization's *best* programmer before PSP training. Of course, the good ones got better also.

Use Two CMMs for Lasting Benefits Sheehan Tells A-SPIN

by Paul Ehrler



Except for a few holdouts in Dilbert-land, most of us know that *CMM* refers to the Software Engineering Institute's Software Capability Maturity Model.

Less well known are the other members of the CMM family—the People CMM, the Systems Engineering CMM, the Software Acquisition CMM, and the Integrated Products Development CMM. In a similar vein, a group at the University of Illinois has introduced a Testing Maturity Model.

What's the justification for this flourishing cottage industry?

Dangers of a la Carte

For starters, with so many models, an organization can tailor a strategy of process improvement around its own unique set of circumstances. There's no question that this makes much more sense than having a group trying to shoehorn itself into a one-size-fits-all scheme.

If you're not careful, though, too much of an *a la carte* approach to process improvement might cause you to miss out on some major long-term benefits. Such was the message from Carol Sheehan, who presented "The Culture of Process: Synergistic Maturity in P-CMM and SW-CMM" at

the April meeting of the Austin Software Process Improvement Network.

Sheehan, of Austin's Sterling Information Group, began by reciting a few statistics to show the value of a CMM-based process improvement program. Seeing numbers like a 22 percent gain in pretest defect detection or a five-to-one return on investment, one would think that nearly every organization would have embraced the Holy Grail of CMM by now. Alas, that is not quite the case.

Too Narrow a Focus

Many managers have yet to be convinced that the CMM is a good investment. Some who took the plunge were not able to sustain the benefits, and in a few unfortunate instances reverted back to pre-CMM business as usual.

Sheehan believes that many of these failures can be attributed to the fact that organizations often focus too narrowly on the Software CMM and not at all on the People CMM. Those who use both are far more likely to see lasting benefits than those who concentrate solely on the former.

Not only does the P-CMM directly fill in some of the gaps left by the SW-CMM, it also helps avoid misinterpretations of the

Prisoners (continued from p. 7)

Specifically, I'm an INTP, where the "I" stands for Introverted. That might be why I sometimes feel uncomfortable in high-intensity settings where critical evaluation is difficult. Perhaps it's easier for others to assimilate what is being said without being so distracted by the way it's presented, but before the evening was over I wasn't quite sure whether I was hearing about the One True Way or the Opportunity of a Lifetime.

By Whatever Means Necessary

A more troubling aspect of Penny's presentation has less to do with the way it was delivered and more with his underlying assumptions. He strongly believes in the inevitability, almost a desirability, of intense dog-eat-dog competition. His advice is clear: We must be prepared to protect our slice of the pie by whatever means necessary. In my view, a blend of cooperation and caution seems more appropriate, especially if we're

interested in larger issues, such as improving our industry as a whole, not just our company.

Different Strokes for Different Folks

While we may quibble over the details, there is no question that Penny's message is an important one. Even the methods that I criticized are undoubtedly valid for some persons and in some settings.

The important thing is that we agree on what really matters. At the end of his presentation, Penny distributed an evaluation form. Since this probably isn't done very often at revivals, I perhaps ought to temper my earlier comments somewhat. He is obviously eager to learn and improve—almost as though he's one of us!

So Stu, if I seemed a bit harsh on my audience feedback form, that was just a quick first impression. You know how we INTPs are.

latter. For example, consider how the SW-CMM emphasizes organizational continuity after the loss of key personnel. CMM Level 1 is often characterized as "hero based," where the organization is at the mercy of particular individuals.

Overzealous managers who want to avoid this dependency might be tempted to shift towards an atmosphere of interchangeability, where damage to morale could more than offset any benefit.

Using the Two CMMs

Sheehan described two basic approaches to using the two CMMs: either build a foundation with the P-CMM before introducing the SW-CMM, or bring both CMMs to bear simultaneously.

She claims that the two approaches yield comparable lasting benefits. The key concept here is "lasting." Disappointing results

CMM (continued on p. 9)

Software Quality Matters

Editor: Cinda L. Cyrus

© 1998 by The University of Texas at Austin. All rights reserved

Printed in the United States of America

Editorial Board Members

Al Dale	Cinda Cyrus
Jack Odom	Herb Krasner
Joanne Click	

Contributors to this Issue

Mark Azadpour	Carolynn Elmshouser
Celeste Armstrong	Gale Hathcock
Candy Walser Berry	Herb Krasner
Debbie Caples	Mark Paulk
Joanne Click	Marilyn Robertson
Cinda Cyrus	Geree Streun
Paul Ehrler	

Software Quality Matters is published quarterly by the Software Quality Institute at The University of Texas at Austin College of Engineering, in cooperation with the Austin Software Council and the Austin Software Process Improvement Network.

Brand names, company names, trademarks, or other identifying symbols appearing in text and illustrations are used for educational purposes only and do not constitute an endorsement by the authors, editor, or publisher.

For additional information, contact —

Software Quality Institute
The University of Texas at Austin
PRC MER Mailcode R9800
Austin TX 78712-1080
Phone: (512) 471-4875
or: 1-800-687-8012
Fax: (512) 471-4824
E-mail: info@sqi.utexas.edu
WWW: <http://www.utexas.edu/coe/sqi>

Conduct Your FTRs Early, Rigorously, And Methodically during Development

by Celeste Armstrong

One of the most effective activities that a project team can engage in to reduce the number of errors in its work products is a Formal Technical Review (FTR) early, rigorously, and methodically during the development process.

The inspection process is the most formal of all the technical reviews and is valuable because it is a repeatable process, and metrics gathered (consistently) from these reviews can be used to project schedules and determine problem areas.

The learning curve on the inspection process is about a month. Since some of you have already been doing peer reviews, but not formally, the learning curve may vary (upward or downward).

Many of the complaints about the FTR process stem from inexperience, inconsistency in applying the process, and what I generically call *the bad meeting*. If you cannot conduct an effective meeting, then the FTR will be problematic for you. While a bad review is worse than no review at all, a properly conducted review can be the single most effective activity a team can perform to improve its work product.

Transitioning to More Formal Methods

I have worked in organizations that “do reviews,” but frankly—how do I say this diplomatically—they are unaware of the true review process.

As I mentioned, organizations that do not conduct effective meetings in the first place will have particular problems with

CMM (continued from p. 8)

after an initial foray into process improvement will make a decision-maker much more reluctant the next time. Save the high risk for later—after the foundation is in place. The important thing while building that foundation is to get it right the first time.

An Unexpected Bonus

During the Q&A period, we received an unexpected bonus. In the audience was none other than Bill Curtis of SEI, who gave us an extemporaneous insider’s view of the P-CMM. It turned out to be a great complement to a most informative evening.

The P-CMM web page is <http://www.sei.cmu.edu/cmm-p/>

the FTR. Another problem results from prideful teams that think they already know how to conduct meaningful reviews. The bottom line is this: learning the process requires an investment in time, but it will pay off richly in the long run.

So, how do you get an organization to recognize that maybe the reviews it has been performing are not optimal? Or, if your organization is of the open-minded variety, how can you tell whether you are getting everything out of your current FTR process? This assumes, of course, that FTRs are being conducted on a regular basis.

Beyond the basic “Are you conducting regular reviews” (and the follow-up questions: when and how often), one of the ways to assess the level of integrity of a review is to ask the following questions.

1. How many premeeting issues were recorded? (If you can’t answer this question, your team may not be spending time before the meeting in review.)
2. What was the average time each participant spent in premeeting review? (If no time is being spent in premeeting review, then the meeting is wasting a lot of people’s valuable time.)
3. How many participants were involved in the FTR? (If there are more than six, a red flag should go up. This is also a meeting red flag.)
4. How long did the review meeting last? (More than 90 minutes is treading on disaster.)
5. How many errors or defects were discovered during the meeting? (If they can only guess, then they haven’t recorded the errors, and the meeting was a waste of time.)
6. What was the resulting disposition of the work product? (If that disposition wasn’t determined or recorded and signed off on, it will be difficult to track the follow-up activities identified during the review.)

If you don’t get satisfactory answers to these questions, then some training or retraining is in order.

Okay, So Where Do We Start?

The FTR process includes many variations. It is *not* a one-size-fits-all-situations model. The different types of FTRs include

walkthroughs, peer reviews, round-robin reviews, and the formal inspection.

I believe that as a work product transitions from one phase to another, at some point, an inspection is in order. I also think that the inspection process can be used almost exclusively throughout the process. (This is probably an inflammatory statement to some developers!) In any case, the following list contains a bare minimum of the things you’ll need to do to conduct most types of FTRs.

Assign Roles. The roles of moderator, producer, inspector, recorder, and reader in the FTR meeting should be clearly announced, and the persons holding those roles should understand their responsibilities.

Report Metrics. One very important outcome of the FTR is the resulting metrics. These metrics will not only help you determine the effectiveness of your review process, but also will lay the groundwork for you to perform a causal analysis on the source of the defects. It should be noted, however, that *before* your FTR meeting, you must collect the number of issues discovered and the time spent reviewing them.

Use an Agenda and Other Appropriate Materials. The agenda is an important tool for keeping the FTR on track, as well as all other types of meetings. Technical guide sheets and checklists document quality factors and raise awareness of what constitutes a “good” requirement, design, module/segment of code, or document. These factors should be built in from the beginning of each and every review and not unplanned afterthoughts. Guide sheets can be generic or specific to a particular company, but they should always be used in an FTR.

An area often overlooked (or assumed) in the FTR is the level of expertise in the technical content of the review. Never assume you have a group of experts in the room. In fact, it is good to include a junior member of the team who is either auditing or participating in order to learn the process and the product.

Request Preinspection Review Time. Request 60 to 90 minutes for each review preinspection. The review leader, also known as the moderator, should solicit written comments or an accounting thereof

FTRs (continued on p. 10)

Quality is Everyone's Job With Embedded Quality Engineering

by Mark Azadpour



Moving from the traditional concept of quality assurance to a view of quality engineering as an embedded and key part of each phase of a project's life cycle makes quality the job of everyone in the organization.

Although quality activities must be performed during all phases of a project's life cycle, the number and type of these activities depend on the scope of the project and cost considerations. This article presents an overview of these activities and should serve as a starting point for organizations seeking to establish quality measures.

Factors Affecting All Quality Activities

Several dynamic factors affect all quality activities performed during a project. Two of the most important are scope changes and configuration management.

FTRs (continued from p. 9)

to demonstrate that the reviewer has inspected the material before the FTR. The meeting itself is not to be used for discovery.

Train, Train, and Retrain When Necessary! Move towards the inspection process with a seasoned coach present if at all possible. You should continually evaluate ways to improve your review process. Another way to accomplish this is to have a postreview analysis of the FTR meeting to help identify ways the review process could be improved.

Just Do It!

If you give the FTR process a fighting chance you will be on your way to a higher-quality product. I encourage teams to adopt the review process even if they are not operating at CMM Level 3 (the level that peer reviews appear) because this is something a team can immediately do to improve its work product without waiting for management buy-in and process definition.

Taking the words of a famous manufacturer, "Just do It!," I add "and do it right." You'll be glad you did.

Note: For more information, see the FTR web page at <http://www.ics.hawaii.edu/~johnson/FTR>

Scope Changes. Although changes are inevitable during project execution, the scope and frequency of changes should be kept to a minimum, or the success of the project could be jeopardized. According to a study, scope changes range from 10 to 15 percent for projects with fewer than 1,000 function points; for projects with more than 1,000 function points, scope changes are 30 to 35 percent.

Scope changes typically result from poor and unclear requirements, or they represent bells and whistles added to the project. In order to minimize scope changes, the requirements should be as concise and complete as possible. Furthermore, client and end-user participation are crucial during requirements definition so that they are developed with the user in mind. Finally, a change board should manage the scope-change process carefully so that changes and their effects are analyzed carefully.

Configuration Management. The other factor that has a major impact on quality is configuration management (CM). Close version control is especially important during system testing. The role of configuration management is even more obvious for client/server architecture products to prevent a wrong version of the client being tested against the server.

Several steps can be taken to minimize the risk of configuration management problems. The first step is the creation of a strong configuration management team where changes are approved by a change control board (CCB). Another piece of insurance is to ensure that requirement and design documents are kept up to date and have approved changes incorporated in them. This ensures that changes are communicated appropriately, minimizing the chance of not implementing a change.

Retention of previous versions of work products is also crucial. This history prevents changes from becoming circular and helps maintain a common direction for the project.

Quality Activities during Development and Maintenance

A variety of quality activities should be performed during all phases of project development and maintenance.

Requirements Definition Phase. Major project goals and success criteria are established during the requirements definition phase. Other factors determined at this point include the project's process and life cycle activities and the testing tools and methodologies (i.e., unit, integration, and system test plans) that will be used. Project and testing metrics also should be set. Such metrics could include the number of defects found, the number of errors found in phase, and the effort required to fix each bug per phase. Customer satisfaction surveys should also be considered.

Design Phase. During the design phase, completed work products in the previous phase are revised and kept up to date, and various reviews and walkthroughs must be performed to ensure that the team is in agreement with respect to the design of various components. The author of a component can play a vital role in this phase if he or she can take an objective view rather than a subjective one. This point is extremely important if reviews are not to turn into personal conflicts.

Implementation and Unit Test Phase. As work products evolve, they must continue to reflect the direction the project is moving. Following programming guidelines during the implementation phase will help programs have a common and seamless look and feel. The importance of this point is evident during the maintenance phase when the sustaining team takes ownership of the product and needs to fix bugs or trace functionality through the modules.

Establishing clear goals for unit testing helps both the developers and QA know what the goals are. Furthermore, for large-scale projects, the developer of each box should perform white box testing so that he or she becomes completely familiar with the functionality and interdependencies of different modules.

Performing meaningful walkthroughs and ensuring the traceability of unit test cases back to the design documents and requirements are two other important quality activities. Traceability should be examined from requirements through the unit tests as well. As the project progresses through this phase, black boxes for the system integration test plan should be filled in and completed. Toward the end of this phase, complete integration and system test plans should emerge.

Integration and System Test Phase. Determining the entry criteria to the integration and system test phase is critical.

Embedded (continued on p. 11)

More Q&As on the CMM/TQM Relationship And on Nonpaying Customers and Tailoring

by Mark Paulk

Question: *I've been told that most companies decide to produce cheap products or high-quality products (read expensive). But there are some companies that try to sell/produce cheap high-quality products (called in-betweens). How do these in-betweens perform on the market compared to the other two?*

Paulk Answers: Don't assume that all business decisions are consciously made!

Also, don't assume that a Civic is a low-quality car just because it's cheap. It's a high-quality car (remember fitness for use!) targeted to an inexpensive market. Quality does not equal luxury.

This is the classic misunderstanding of the quality movement. Before studying the software process improvement world, study the TQM world. Read Deming and Juran. SPI is an application of those more general concepts in the software world.

Question: *I think that Levels 2 and 3 are quite different/separate from Levels 4 and 5. My view is that Levels 2 and 3 are parts*

Embedded (continued from p. 10)

For example, what percentage of unit tests should be passed and how stable do the components need to be before the product can be handed over to the test team?

This is one phase where quality is often ignored or viewed casually during crunch time, though such an approach could make the system test and the number of iterations of integration testing very long and tedious. In fact, the success of the project could be jeopardized if the product is sent to integration and system testing before it is stable.

The following need to be considered during this phase.

- Functionality testing
- User interface and human interaction testing
- Security testing
- Performance and stress testing
- On-line help and adherence to actual functionality
- User manuals and how closely they relate to the actual product

Once all parties have accepted the results of system testing, the product should be turned over for acceptance testing.

of a larger goal, namely to stabilize the process(es), to minimize the variance/deviation/fluctuations.

Levels 4 and 5, on the other hand, deal more with adjusting "the level/height of the curve." This is just my opinion. Is it totally wrong? (Doesn't this imply that defect prevention is misplaced at Level 5? I don't know enough of the theory. I have lots and lots of reading to do. . .)

Paulk Answers: Think of it as qualitative vs. quantitative process control then process improvement, which gives you four levels above Level 1.

This means that you disagree with the basic principle of TQM. While I don't particularly mind, it's a very old argument—that quality costs—one that's been debated by far abler folks than I.

I don't disagree. But, I think (it's just an uneducated opinion) it's reasonable to assume that the Levels 2 and 3 block will give a better return on investment than the upper block. Why? Because it is probably easier and cheaper to remove the first errors.

Pick the low-hanging fruit, as we say in the States.

Again, this goes back to the old quality-costs debate. We're seeing so much of this resurface under "good enough software." I have no problem with this concept so long as you realize this is a fitness-for-use debate. When it becomes operationalized as "whatever junk we can get a customer to buy," I think the market will determine pretty effectively.

Question: *I think I'm getting your point, but I'm not sure. Please, could you elaborate on this a bit? I am having trouble remembering all the stuff all the time.*

Paulk Answers: Here's an analogy: GM said it built good cars and could point to domination of market in the '50s through early '70s as proof that they understood what was good enough. Then the environment changed, and the companies that had focused on quality made enormous inroads.

Question: *Here's a new one. Can an organization be rated Level 3 if a project does not comply with Level 2 or 3 KPAs if its customer has specifically said he doesn't want to pay for them? This is in a defense, contract-based environment.*

More Q&As (continued on p. 12)

Information Sources for Local Groups of Interest

Austin Forum for Object-Oriented Technology (AFOOT)—Phone: (512) 452-9455; E-mail: afoot@twr.com

Austin Software Process Improvement Network (A-SPIN) E-mail: gereestreun@earthlink.net

Austin Software Council (ASC)—Phone: (512) 305-0035

Java Klatch—Phone: (512) 258-8437; E-mail: java-klatch-info@iconcomp.com

Acceptance and Product Release Phase.

During the acceptance and product release phase, the functionality of the system should be verified to ensure that all customer requirements have been met. All the test logs, bug reports, and release notes should be prepared. If a separate sustaining team will be inheriting the system, it should be involved at least by this point.

Holding a post mortem and lessons-learned meeting once acceptance testing is completed is another important quality activity and one that should identify improvements for future projects.

Sustaining Phase. Product shipment is the entry criterion to the sustaining phase. Dur-

ing this phase, the test facility should remain available so that regression testing as well as bug investigation can be performed easily.

As suggestions from users are received, they should be collected and reviewed periodically by the review board. This board should decide either to incorporate the changes as a minor release or to roll them over for the next major release. This determination should not be based solely on the programming effort involved but on the system impact and the amount of testing required to ensure the integrity of the system. Thus, version control and configuration management continue to play a critical role even after product release.

SQI Calendar—Meetings and Conferences of Interest

June 18, 1998

A-SPIN monthly meeting
The Perils of Pauline
6:30 to 9:30 p.m.
Pickle Research Campus, Austin TX

June 28–July 1, 1998

IEEE International Conference on
Multimedia Computing and Systems
Renaissance Austin Hotel, Austin TX

July 8–9, 1998

SQI Seminar
Software Project Planning and Tracking
Pickle Research Campus, Austin TX

August 10–11, 1998

SQI Seminar
Software Risk Management
Pickle Research Campus, Austin TX

August 20, 1998

A-SPIN monthly meeting
Keeping Risk and Opportunity in Harmony
6:30 to 9:30 p.m.
Pickle Research Campus, Austin TX

September 11, 1998

Personal Software Process training begins
Pickle Research Campus, Austin TX

September 17, 1998

A-SPIN monthly meeting
Reasons People Do Not Do as Expected
6:30 to 9:30 p.m.
Pickle Research Campus, Austin TX

October 15, 1998

A-SPIN monthly meeting
The Five Responsibilities of Leadership
6:30 to 9:30 p.m.

Pickle Research Campus, Austin TX

November 19, 1998

A-SPIN monthly meeting
Reinforcing Desired Behavior
6:30 to 9:30 p.m.
Pickle Research Campus, Austin TX

February 22–24, 1999

First Working IFIP Conference on
Software Architecture
San Antonio TX

One can always argue that different key practices may be needed and may be appropriate, but organizations that have tailored the CMM for small projects observed that more than 90 percent of the key practices carried forward into that environment. Where you really get large-project specific or organization specific is in the sub-practices. What is important is mapping the concepts and roles of your environment to the concepts and roles in the CMM. Once you establish that relationship, then analyze the applicability of the practices.

Send Your Questions

You are welcome to send me your questions about the CMM. I prefer to receive questions in writing by e-mail. Your name and the name of your company will not be used when I distribute the answer.

Mark Paulk
Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213-3890
Fax: (412) 268-5758
E-mail: mcp@sei.cmu.edu

More Q&As (continued from p. 11)

Paulk Answers: Yes, if it's a rare occurrence, but I would want to see a directive from the customer saying that they refuse this functionality and accept the risk that it entails. More generally, I would expect a company to do the right thing even if the customer says don't do this—to the extent that a supplier would charge *more* for not doing SCM than for doing it (after all, the reasons for the KPAs are driven by business concerns).

Question: *We have been trying to understand the applicability of the CMM to short-cycle-time software support activities. We concluded that applying the CMM in all its rigor will add to the cycle time of the support activity and lead to possible customer dissatisfaction. I am now thinking*

of formulating a support maturity model with the same five levels as the CMM but with different key practices and KPAs.

Are there some guidelines to selecting KPAs and KPs for a maturity model? Is there a generic process for this? Are there any references available? Any light you can throw on this will be deeply appreciated.

Paulk Answers: A report by Mark Ginsberg on tailoring the CMM should help. Check the anonymous ftp site for it (ftp.sei.cmu.edu, pub/cmm or pub/documents).

I disagree with the idea of having different KPAs and claiming it's the same maturity levels. I strongly argue that all of the KPAs, with the exception of subcontract management, apply in any software development or maintenance environment, while the implementation may differ dramatically.



Software Quality Institute
The University of Texas at Austin
PRC MER R9800
Austin, Texas 78712-1080

Nonprofit Org.
U.S. Postage
PAID
Austin, TX
Permit No. 391