

Software Quality Matters

- Volume 5, Number 2, Fall 1997

Articles in this issue:

A Graduate Speaks Out: **Thinking I Had Little to Learn from the SWPM Program, I Got an Education; You Should Too**

Alabama to Host 7ICSQ in October

CoSQ Part III: **Combining CoSQ & SPI Shows Value of Quality Initiatives and Yields Significant Cost Savings**

Book Review: **You Can't Always Judge A Handbook by Its Cover** by David Rentschler

The Key to a Winning Business Plan: Inform and Excite the Investor Quickly by Gene Lowenthal

Maturing Austin Software Council Celebrates Year of Achievements

Using Factual Data to Change the SQA Role from Bad Cop to Value-adding Team Member by Herb Krasner

Belated Congratulations to SWPM Grads: Armstrong, Rubinstein, Neidhart, Tabrizi!

Measurements Used for CMM Also Indicate Project Success
by Paul Ehrler

Transforming a Sow's Ear -- Documentation: Engineer's Curse, Editor's Delight by Jim Terrel

Questions & Answers on the CMM/TQM Relationship by Mark C. Paulk

Java Klatch/AFOOT Joint Meeting Slated

SQI Calendar: Meetings and Conferences of Interest

A Graduate Speaks Out

Thinking I Had Little to Learn from the SWPM

Program, I Got an Education; You Should Too

by Celeste Armstrong

When I first heard about SQI's certificate program for software project management (SWPM), I was very envious of all the graduates who had gone before me. Then I talked to a few of them and became a bit disillusioned because, really, I HAD already learned most of the things they listed as being in the 6 modules. Managing Technical People (done it), Initiating the Software Process (ditto), Elements of . . . (yawn).

Still the notion of actually having competency-based exams and getting certification. THAT was intriguing. So, I pitched the case for taking the class to my manager, he obliged, and the rest, as they say, is history.

My Eyes Started to Open

To say that I began my journey a little prideful would put it mildly. In the first SWPM module, "Managing Technical People," we talked a lot about personalities of IT professionals and even did a Myers-Briggs personality typing. Well, big deal. I knew my personality type, and I was beginning to think I should skip class! The psychologist who led the personality typing discussions was dynamic and interesting, however. We did a few very interesting class exercises, and, yes, my eyes started to open.

I went back to my work team, which consisted of 12 quality engineers, and we talked about Myers-Briggs personality types and how we could understand our own and each other's types **and this part was new for me** *how we could leverage personality types to get our jobs accomplished more effectively*. Not only that, when later I was teaching a course entitled *Software Engineering Basics*, I incorporated some of the personality type analysis in the framework of developing requirements. It was great fun and very enlightening for some of the participants.

Not only did SWPM Module 1 help me understand myself and my colleagues, the module gave me information that I was almost immediately able to share with my organization. I began to think that I COULD get something out of this class after all!

Getting out What You Put in

I quickly learned during that first module that, as in most things in life, you get out of the program what you put into it. The quality of the SWPM course was closely related to the quality of individual participants and their teams. That explained some things for me, like, for example, how a graduate of this program could possibly have not improved his work performance as a result of completing the course.

Our teams were groups of three to four participants. Each team completed the assignments and exercises together, giving us the sense of a real-world working environment. As is always true, some teams were more effective than others, and some individuals were more effective than other individuals.

I was fortunate to be on a team with people who were really invested in learning what the assignments had to offer. As a result, I believe we derived the maximum benefit from the course because of what we were willing to put into it. Conversely, teams and individuals who were not so diligent missed some or a great deal of what was available to learn.

The Homework Really Applied

I found that homework assignments were pertinent and could be related back to my actual work environment, either directly or indirectly.

It became clear that the value of the program was not only the content but that, in applying the information learned and homework activities to daily work, the benefits were multiplied. Not only that, I was able to brainstorm with other members of the class and learn from THEM also! So, my initial pride about already knowing it all quickly began to fade.

Alabama to Host 7ICSQ in October

It's not too late to make plans to attend the 7th International Conference on Software Quality (7ICSQ) in Montgomery, Alabama, on October 6-8, but you'd better get a move on it.

7ICSQ will offer full- and half-day tutorials on Monday, October 6, 1997, followed by a two-day technical program. Post-conference courses are planned for October 9-10.

The conference is sponsored by the American Society for Quality (ASQ); Auburn University-Montgomery, Information Systems and Decision Science Department; and the Montgomery Area Chamber of Commerce, Information Technology Committee.

Tutorials

The following full-day tutorials are planned.

- Guide to Software Measurement Startup
- Requirements Testing: the Basics
- Charters and Chartering in Software Process Improvement

Half-day tutorials include SPICE and SPICE-Compliant Assessment, Implementing Value Added Audits for Software Process, Creating a Software Engineering Culture, Project Measurement Methodology, Process-based Product-focused Risk Assessment, and the Capability Maturity Model for Software: Version 2.

Technical Program

Four tracks will be offered each day of the technical program.

On October 7 you may choose from Process Improvement, Management Issues, Assessment/Auditing, and Metrics. On October 8 your choices are Metrics/Measurement; Inspection, Test, and Quality Characteristics; Governmental and Military; and Special Topics.

Austin's Herb Krasner, the conference's keynote speaker, will discuss Software Cost of Quality.

Additional Information

To receive a copy of the conference brochure, contact ASQ Headquarters at (800) 248-1946 and ask for Priority Item B0507 or look for more info on the 7ICSQ home page at www-biz.aum.edu/~tgriffin/7icsq/

CoSQ Part III

Combining CoSQ & SPI Shows Value of Quality Initiatives and Yields Significant Cost Savings

by Dan Houston and J. Bert Keats

In parts I and II of this article, which appeared in the last two editions of Software Quality Matters, the authors identified the cost categories associated with the cost of quality (CoQ) and discussed the relationship between the costs of achieving quality and costs due to lack of quality. They went on to examine the elements of the emerging field of the cost of software quality (CoSQ) and the potential benefits of employing it, as well as CoSQ's use in software process improvement.

Now, in Part III, the final installment of this article, Houston and Keats discuss how the optimum cost of quality is shown as absolute costs against a quality measure and how simulation can be used to predict CoSQ for software process improvement (SPI).

Predicting CoSQ for a Process Improvement

Many improvements can be applied to any given process, so knowing the probable effects of a particular process improvement before its implementation helps clarify which improvement to apply.

The effects of a process improvement may be ascertained through the experience of other organizations (usually in published studies) or through pilot studies. Software processes vary significantly from one organization to another, however, and pilot studies may be expensive to implement and difficult to control for the variable under study.

Another way to learn the effects is to use software process simulation to isolate the effects of a single process improvement.

Using Software Process Simulation

Our study employed a dynamic simulation model to compare (a) the CoSQ effects of developing a product when SQA reviews are employed with (b) the CoSQ effects of developing a product when formal inspections are added to the process.

The model, which was developed by Tvedt (1996), is based on Forrester's System Dynamics Modeling approach (1961) and builds on software simulation work by Abdel-Hamid (1991). (Abdel-Hamid used a system dynamics model to demonstrate that optimal SQA expenditures range between 10 and 30 percent of development effort.)

The model covers development phases from requirements definition through system testing, providing for 116 inputs and 110 outputs. Most of the inputs, such as work rate and error correction rate, are for the sake of calibrating the model to a particular development organization's process. The other inputs provide for characterizing a project by, for example, product size and the numbers of new and experienced engineers.

For this study, the model was calibrated for an 80,000 LOC COBOL project produced in a local software development organization using 15 full-time engineers and a waterfall process. The first series of runs modeled a process without formal

inspections; the second, with formal inspections. In each series, the amount of testing was varied to produce a variation in the number of product defects at the completion of system testing, the study's quality metric.

Prevention costs were the hours spent in SQA reviews/inspections. Appraisal costs were the hours spent in testing. Internal failure costs were the hours spent in rework. External failure costs were calculated from the number of defects in the product at the conclusion of system test, estimating that the cost to fix an external defect is twice the cost to fix an internal defect in system test rework for a small project (Boehm 1981). (This ratio can be ten or higher on large projects.)

Figure 5 shows the CoSQ for the simulated development without inspections. Figure 6 shows it with inspections. CoSQ is shown as absolute cost to illustrate the optimum total CoSQ (TCoSQ).

This study shows that the most important result of adding inspections to the development process was a decrease in the TCoSQ: the optimum fell from 3500 hours to 2900 hours, indicating a potential savings of 600 development hours for this project.

Another result is that the optimum TCoSQ for this project shifted from 2.7 defects per KLOC to 1.6 defects per KLOC with the addition of inspections. Adding inspections to the development process shifted the optimum TCoSQ significantly in the direction of higher quality.

Knowledge of the optimum TCoSQ serves the utilitarian view of quality, which is appropriate for many kinds of commercial software (Bach 1995).

The variance in CoSQ in this case primarily depends on the amount of testing and the cost of that testing. This study suggests that appraisal costs begin to increase significantly when the product's defect rate falls to between 1 and 3 defects/KLOC. This is consistent with Krasner's study of space shuttle onboard flight software (Krasner 1994) in which the per-defect cost of testing increased dramatically when the product defect rate fell below 1 defect/KSLOC.

Conclusions

CoQ is a proven technique in manufacturing and service industries both for communicating the value of quality initiatives and for indicating quality initiative candidates. CoSQ offers the same promise for software process improvements, but it has seen little use so far in the software development community.

Initial uses of CoSQ show that it is a very large percentage of development costs, 60 percent and higher for organizations that have not yet undertaken process improvement programs. CoSQ use can, however, demonstrate significant cost savings—such as Raytheon Equipment Division's fourfold reduction in rework; for software organizations willing to undertake a process improvement initiative. In addition, CoSQ can be used to identify candidate process improvement initiatives; it provides a means of distinguishing areas where quality costs are high and a means for estimating the potential cost benefits of process improvements.

© 1996 by Dan Houston and J. Bert Keats.

Note. Bibliographic information may be obtained from the authors at dan.houston@iac.honeywell.com

About the authors. *Dan Houston is a software engineer at Honeywell and an Industrial Engineering Ph.D. student at Arizona State University. His research interests include dynamic process modeling, quality management, and process improvement.*

J. Bert Keats is associate professor of Industrial and Management Systems Engineering at Arizona State University. His professional interests include reliability engineering, statistical process control, and the software quality audit. He was a 1994 co-winner of the Brumbaugh Award for a paper published in an ASQC journal that made the greatest contribution to the industrial applications of quality control.

Book Review

You Can't Always Judge A Handbook by Its Cover

by David Rentschler

I heard about the *Handbook of Software Reliability Engineering* about a year before it was available. Being a practitioner in the field of software reliability engineering (SRE) and an engineer familiar with and comforted by engineering handbooks during and after my college days, I eagerly awaited its arrival.

Handbook vs. Textbook

Unfortunately, I was disappointed when the book arrived. For what I received was not a handbook, as the title promised, but a

textbook, a very good textbook.

What's the difference you ask? A handbook provides answers. A handbook presents accepted methods and works out solutions to basic problems and their variations. To me, handbooks are easy to understand, and, most importantly, they bring the mathematics to a level that does not require a refresher in calculus or advanced statistics.

When you're under pressure at work to provide a solution to a tough problem, the last thing you want to see is a math formula that screams (like the familiar cartoon) <<then magic occurs here.>> The *Handbook of Software Reliability Engineering* is a very good textbook. It provides some of the latest thinking about software reliability practices, however, and the end of each chapter asks a lot of good questions that are left as an exercise for the reader to answer.

An Overview of the Contents

The *Handbook of Software Reliability Engineering* has three major parts, organized like a tree. Technical Foundations, the first part, is just that; it is the trunk of SRE. Practices and Experience, the second, adds to and broadens the scope of SRE; this part forms the major limbs. The third part, Emerging Techniques, branches out and is supported by the first two parts.

Each part is authored by a respected leader in the field, sort of a who's who in software reliability.

Technical Foundations. The first part, Technical Foundations, provides an overview of the basics of software reliability practice. (Before you begin, you may want to consider reading through the roots of the tree: Appendix B, Review of Reliability Theory, Analytical Techniques, and Basic Statistics.) The Technical Foundations part contains Software Reliability and System Reliability, Software Reliability Modeling Survey, Techniques for Prediction Analysis and Recalibration, and Operational Profile.

Practices and Experience. The second part, Practices and Experience, provides more applicable information to everyday software reliability work. Though an understanding of the technical foundations of the first section is helpful, it is not necessary. The Practices and Experience part contains Best Current Practice of SRE, Software Reliability Measurement Experience, Measurement-based Analysis of Software Reliability, Orthogonal Defect Classification, Trend Analysis, and Field Data Analysis.

Emerging Techniques. The third part contains some interesting articles about the trends in software reliability work. There are some good ideas here, and I am especially intrigued with the simulation chapters. The Emerging Techniques part contains Software Metrics for Reliability Assessment, Software Testing and Reliability, Fault-Tolerant Software Reliability Engineering, Software System Analysis Using Fault Trees, Software Reliability Simulation, and Neural Networks for Software Reliability Engineering.

Bonus: a CD-ROM of Tools

Rounding out the book, and one of its benefits, is a CD-ROM containing several software reliability programs. These are good tools to aid in software reliability analysis: AT&T's SRE Toolkit (PC); U.S. Navy's (now public domain) Statistical Modeling and Estimation of Reliability Functions (SMERFS) (PC); The UK's Reliability and Statistical Consultant's Statistical Modeling and Reliability Program (SRMP) (PC); LAAS of France's Software Reliability Program (SoRel) (Mac); JPL's Computer-Aided Software Reliability Estimation Tool (CASRE) (PC), and Bellcore's Economic Stop Testing Model (ESTM) Tool (workstation).

Other SRE Books

I can also recommend another couple of recent SRE books.

- *Ensuring Software Reliability* by Ann Marie Neufelder, Dekker, 1993
- *Metrics and Models in Software Quality Engineering* by Stephen H. Kan, Addison Wesley, 1995

Still Waiting for a Real Handbook

As the field of software reliability matures, I believe there will be a need for a real handbook. The kind with answers. The kind where the solution methodology becomes practice and is referenced. The kind that comforts when the going gets tough. If you find one, let me know.

About this book. *Handbook of Software Reliability Engineering*: Michael R. Lyu, editor; IEEE Computer Society Press; McGraw-Hill, 1996

About the author. David Rentschler is a software reliability engineer at Tandem Computers in Austin. He has more than 15 years' experience in software development, process, and quality improvement. He has a BS and MS in Industrial Engineering from Texas Tech and

is a Registered Professional Engineer in the State of Texas.

The Key to a Winning Business Plan: Inform and Excite the Investor Quickly

by Gene Lowenthal

Entrepreneurs tend to be concerned about what needs to be included in the business plan. While this is proper, the overriding concern should be whether the document teaches and sells effectively.

I see too many plans that seem to have all of the required content but don't do a good job of explaining the business or getting me excited about the opportunity.

The business plans I'm talking about are designed for potential investors (institutional venture capitalists or individual

Business angels: This plan is very different from a business plan for internal consumption.

The typical venture capitalist (VC) sees dozens of business plans a month, and only a very few of them will get a second look. If you're writing a business plan you must be keenly aware that you are competing with other hungry entrepreneurs.

Assume that the VC will read as little as possible of your plan during the screening process. Specifically, he or she will only read the executive summary and the financial projections.

In this respect, the executive summary *is* the business plan. The executive summary, which is the first section of the plan, presents your plan in a nutshell. It should be brief (2 to 5 pages), and it must be exceptionally clear and compelling. Otherwise, the game is over.

The A+ Executive Summary

Writing an A+ executive summary is a combination of skill and art. In essence, it is a very good marketing piece about the company but without the gratuitous hype. Remember, you are addressing a savvy, sophisticated, and somewhat cynical audience.

Here are the most important topics to address in your executive summary.

- **What:** what is the product or service? What key technologies are related to your product or service?
- **Why:** why will people buy your product or service? What important problem are you solving? What pain are you removing? What vacuum are you filling? What value are you creating for the buyer?
- **Who:** who are the key players on the management team and the board of directors? Outline their backgrounds in a paragraph or two. Investors need to see a strong team or help build one. They bet on people much more than they bet on ideas or technology.
- **Size:** what is the size of the opportunity? Investors need to know that you are attacking a large market (measured in hundred of millions of dollars per year) and that you plan to grow a large company, in excess of \$50 million after a five-year period from inception.
- **Market positioning:** what is the shape of the competitive space? What are the categories of competing products or services? How are you positioning your offering within that space? Most important, what is your *sustainable competitive advantage*? What makes your offering different and better than a customer's alternatives now and in the future? What barriers to entry will you be creating? It is in this area, market positioning, that 90 percent of all business plans fail. It would be wise to have the assistance of a marketing strategist who is intimately familiar with the target market.
- **Status:** how far along are you? Is the product developed? Is it in the market? Who are your key customers or partners? How big is the company in terms of people and revenue?
- **Financial summary:** what is the business model? In what ways does the business activity generate revenue?

How much money is the company looking for? What are your historical and projected revenues and earnings before taxes (EBIT) for the next three to five years? This can be stated as a simple two-row matrix where the first row is revenue, the second row is EBIT, and the columns represent years.

A Methodology for Creating the Executive Summary

Begin creating your executive summary by trying to explain your business in less than a minute to an outsider. By an outsider I mean someone who, like a VC, knows about the software world in general but is not a specialist in your particular domain. Imagine that you have only the amount of time of one elevator ride to get the VC's interest. Keep refining your elevator speech until it works and the outsider, so to speak, gets it.

Next, develop a 20-minute slide presentation that explains your business in somewhat more detail. Keep tuning it until an outsider not only gets it but admits to being intrigued by the opportunity.

Then, bring in some really knowledgeable mentors who will provide valuable feedback not only about the quality of the presentation but about the strategy itself. At this point you should have a cogent model of the business in your own head.

Now and only now can you start to write a winning executive summary.

The Rest of the Plan

Oh, don't forget to write the rest of the business plan! The most essential section includes the financial projections. These must show the desired revenue growth while remaining realistic with respect to a number of parameters such as growth rate, profit margins, and future investment needs. If you are not versed in such matters, get help.

About the author. *Gene Lowenthal, a partner in Growth Capital Partners, an investment banking firm, has spent 25 years in high tech in Austin. A member of SQI's Advisory Board, he has also spoken about business plan development to many groups, including the Austin Software Executives' Group, the IC² Institute, and the Texas Capital Network. He received his Ph.D. in Computer Science from UT.*

Maturing Austin Software Council Celebrates Year of Achievements

It has been a busy year for the maturing Austin Software Council (ASC), and, according to Cerise Blair, ASC director, the organization has met several significant goals in its quest to expand networking and resource services to Austin area software professionals.

ASC Web Site

One of ASC's most far-reaching achievements is the introduction of the new ASC web site, which recently moved to the Quadralay server. If you haven't checked us out at www.austinsoftwarecouncil.org we hope you soon will, Blair said.

"Our site offers meeting notices, details about membership, upcoming events, a human resource directory, office listings, and more. Soon you'll see a list of ASC member companies, and, in the near future, members will have access to full member directory through use of a password," Blair added.

President/CEO Peer Group

After months of research and planning, ASC achieved another goal in March with the launch of the first President/CEO Peer Group Luncheon for high-level software executives. The group meets every four to six weeks to exchange views and explore avenues to address needs.

"We are also currently developing groups for seasoned marketers, executives of mid-size software companies, and entrepreneurs with young companies," Blair said.

Silicon Hills Source

April saw the birth of Silicon Hills Source, a high-tech database containing information on more than 1700 companies. Designed by ASC's past Vice Chair, Gene Lowenthal, the database was produced by a partnership composed of ASC, The Austin Business Journal, The American Electronics Association, and Civic Pride. All four groups shared in the data collection. The database was developed and is maintained by Civic Pride.

"The address for the database is www.siliconsources.com and if you want to print reports and manipulate data, you should contact Civic Pride for membership details. The *Austin Business Journal* will offer a print version in October. Interest in the site is clear: it is receiving 100,000 to 300,000 hits per month," Blair said.

ASC-sponsored Conferences

ASC and Gene Lowenthal also worked together to capture the 1998 IEEE MultiMedia Conference for Austin. The event is expected to bring several hundred attendees and a nice economic boost to Austin next year.

"The Texas Software Symposium (TSS), which we sponsored in May, was a great success," Blair said. "It featured 20 top names in the software industry and attracted 230 attendees." Mike Maples, who retired from Microsoft to move to Austin, was keynote speaker. Panel discussion topics included marketing, development, legal issues, and human resources. The conference ended with top executives Bill Bock, Robert Fabbio, Frank King, and Steve Papermaster presenting refreshingly candid 20/20 hindsight views of what they'd do (and not do) again in their careers. "Reviews of TSS have been wonderful, and we plan a second TSS in 1998," Blair added.

Additional Information

For more information about ASC activities and opportunities, phone (512) 305-0035 or contact Cerise Blair at cerise@ati.utexas.edu

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: tn01022@psinet.com

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

E-mail: cerise@mail.utexas.edu

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

Using Factual Data to Change the SQA Role from Bad Cop to Value-adding Team Member

by Herb Krasner

Holding software quality assurance (SQA) responsible for the level of quality in a software product is like expecting the finance department to be responsible for the company's profit margin.

Ultimately, the whole development team is responsible for the quality of the software produced.

Within the development process, the role of SQA has traditionally been that of bad cop, checklist executor, document reviewer, and/or process auditor. For example, sometimes SQA engineers reviewed and approved documentation before it was baselined by software configuration management; but not necessarily for quality content.

The idea that the quality of a software system is governed by the quality of the process used to develop and maintain it is supported by the Software Engineering Institute's Software Capability Maturity Model (CMM) Level 2 key process area (KPA), SQA.

Following are the goals of the CMM SQA KPA.

1. SQA activities are planned into the project.

2. Adherence of work products and processes (software, documents, etc.) to standards and other requirements is objectively verified.

3. Affected staff are informed of the status and results of SQA activities.

4. Noncompliances that cannot be resolved by the project are addressed by senior management

All four of these goals can be accomplished within the development processes of the project team as long as (1) team members understand and play their "objectivity" roles and (2) there is a mechanism for escalating serious noncompliances issues beyond the project management chain of command.

What is the Role of SQA?

It is a shame that many SQA personnel have been relegated to performing the role of process bad cop and/or external auditor. In many situations this creates an adversarial situation that is often counterproductive to achieving project success. Wouldn't it be nice if SQA could actually perform its desired role of assuring the quality of produced software?

The main problem is that the role of SQA in many companies is very confused. Why else would some companies hire entry-level testers who can only be checkers of the obvious? This was Mark Azadpour's complaint in his SQA article that appeared in the last issue of *Software Quality Matters*.

SQA should primarily be involved in such activities as:

- quality goal and metrics establishment,
- requirements validation,
- test planning,
- test environment setup,
- test database development,
- participation in design/code/document reviews, and
- the various phases of a structured test process (i.e., a defined approach to unit testing, subsystem testing, regression testing, integration testing, and operational and acceptance testing).

It is not clear whether SQA should be performing project process definition and process adherence determination or whether these activities should be shifted to a software engineering process group (SEPG) function.

An Enlightened Approach

In a better world a process improvement coach would assist software engineers in merging SQA techniques into their peer review, inspection, and related processes. Once these skills become institutionalized, most SQA tasks can become part of the normal everyday software development processes.

As teams begin to move from an SQA mentality to a quality management mentality, quantifiable and measurable goals for software and process quality will begin to drive the development team to success.

Example goals that might be adopted include the following.

- Software quality
 1. No serious defects delivered (where serious is precisely defined)
 2. Defect density at less than 1 defect per million lines of code
 3. Reliability of better than 10^5 hours for MTTF
 4. All priority-one features included and functional
- Process quality
 1. Defect phase containment at 95 percent effectiveness
 2. Rework at less than 20 percent of total development effort
 3. 75 percent of all defects detected prior to testing

What is the Value of SQA?

One of the more difficult tasks faced by any SQA organization is a quantitative understanding of the costs and benefits of its function. Without well-defined software quality goals and metrics, this determination is nearly impossible.

We can look at the value of SQA by asking about how much defects (or problems, changes, bugs) cost to fix. By tracking defects and the cost of fixing them, we should be able to show that the cost of SQA work really pays off when defects can be prevented from being delivered.

A major item in the value of SQA is in the cost difference of identifying and fixing defects before the software gets shipped versus the cost of handling and fixing an externally reported problem/defect.

Belated Congratulations to SWPM Grads: Armstrong, Rubinstein, Neidhart, Tabrizi!

Celeste Armstrong, Viviana Rubinstein, Chuck Neidhart, and Ali Tabrizi: please accept our belated congratulations for successfully completing all courses in SQI's Software Project Management Program and receiving your certificates. We salute your perseverance and hard work.

Our omission of your names from the listing of graduates in the last issue of *Software Quality Matters* was inadvertent, and SQI sincerely regrets the error.

Measurements Used for CMM Also Indicate Project Success

by Paul Ehrler

Addressing a summer meeting of the Austin Software Process Improvement Network, Gary Cobb convincingly demonstrated how some of the same measures that help gauge an organization's CMM maturity level are directly related to bottom line indicators of project success.

A member of SQI's Advisory Group, Cobb is also a systems/software quality assurance manager for information services at Dell Computer and director of Southwest Texas State's multimedia lab.

Speaking on the topic of "Beneficial Software Measurements," Cobb used SEI's five-level CMM, which relies heavily on process measurement, first as a means of establishing a process baseline, and, in particular, as a central Key Process Area component at Levels 3 and 4.

No Hard and Fast Rules

Unfortunately, gathering and using software metrics is not as straightforward as one might hope. Even so-called intuitive measures can be tricky when it comes down to actual data collection, Cobb said. For example, the notion of defect does not remain constant across all project phases, and this dependence must be reflected in the data being collected.

Citing one project that used twenty-one distinct categories of defects, Cobb said this is typical. As one might expect, there are no hard and fast rules: each organization must determine what basic measures should be made for a particular project and how they should be collected.

Multiple Uses of Software Metrics

Software metrics go far beyond determining whether a completed project is a success or failure, of course. They are used to help plan courses of action by comparing estimates to predetermined threshold values. By highlighting critical deviations before a project slips into crisis mode, a well-designed system of metrics can become the canary in the coal mine.

Another use of metrics is to dynamically fine tune estimates of critical indicators, Cobb said. For example, trends in defect density, starting in the early stages, can be extrapolated to successively narrow the estimate of final product quality, cost, or time of completion when it comes down to actual data collection.

Combining Various Measures

In order to take best advantage of software metrics, there needs to be a way of combining various measures to form reliable conclusions. The Rayleigh curve, a fairly simple, straightforward mathematical relation, is quite useful for this purpose, Cobb explained.

Barry Boehm's COCOMO is an example of a predictive model based in part on the Rayleigh curve. Since the Rayleigh curve is not one but a family of mathematical relations, parametrics must be carefully chosen. Predictive models such as COCOMO use empirical data to establish these values.

Different Models, Different Predictions

Different models can easily lead to very different predictions for any given project, and, according to Cobb, it is not always obvious in advance which one is most appropriate. Even if one model stands out, it would be nice to be able to tweak it somehow to make it even better.

Fortunately, Cobb added, new software tools in effect use data collected in early phases of a project and on earlier projects to generate custom models. One such tool is the Software Error Estimate Program (SWEEP), which was developed by the Software Productivity Consortium.

With experience, SWEEP and other tools can demonstrate remarkable predictive power and help us move in the right direction toward process maturity, Cobb concluded.

Transforming a Sow's Ear

Documentation: Engineer's Curse, Editor's Delight

by Jim Terrel

Most of us have experienced projects with hard deadlines and enough schedule slips to devour the planned time for packaging and documenting our work. We are already completely burnt out by the tasks we've actually been trained to do, such as design and coding, but now we must produce the

documentation.

Ah, documentation . . . the software engineer's favorite activity, the maintenance programmer's next to last hope, the poor user's final hope, and: with the exception of rational planning: the activity most likely to be given short shrift in any project.

So we found out what we call "draft documentation," often totally unaware of our use of implicit knowledge: that is, referring to things within our heads but not known by the audience of the document. We then thank our creator for letting us have a life for the brief period until the next project.

But what happens to that pile of mush we call documentation? With luck our project has a technical editor to help us transform the mush into something sharp, crisp, and useful to someone other than ourselves.

Technical editors are often appreciated in the same degree as killer bees and fire ants. Just when we thought we were done, they are bugging us to use the English language, to clarify, to rewrite, and to define mnemonics.

This article is a brief attempt to learn a bit more about technical editing and editors. Cinda Cyrus, editor of *Software Quality Matters*, has agreed to answer a few questions concerning her area of expertise.

Question: *How did you come to be the editor of Software Quality Matters? How long have you been producing it and what's your background?*

Cyrus answers: I seem to have arrived at the hospital at almost the minute the Software Quality Institute was born, thanks to an invite from Herb Krasner. The idea of *Software Quality Matters* sprang up almost immediately, and he and Joanne Click and I brainstormed the first edition into being in fall 1993, and I've been producing it ever since.

I've spent some 30 years (yes, I began working at age 4) in technical writing and editing in Austin, but software is only my latest incarnation. Perhaps you've read some of my scintillating petroleum textbooks: *Well Service and Workover? Secondary Cementing?* No? You're probably waiting for the movies. I know I am.

Anyway, I was introduced to the software world at Lockheed, and now I'm working in software quality engineering at Tracor.

Question: *Assuming the axiom "If it ain't broke, don't fix it!" applies to editing, how do you know something is broken and when you've fixed it?*

Cyrus answers: My first clue that something I'm reading needs fixing is when I say, "Huh?" Then I stop and figure out what the author is trying to say.

Sometimes the only equipment I need is the old syntax polisher and the 000 sandpaper. Other times the eggs are scrambled and I have to reconstruct. I also keep a chain saw, but I don't have to use it very often.

The figuring out part is the challenge. The fixing is the fun. It's rather like bridge, the bidding and the play of the cards. Like any skill, the longer you practice, the better you get. Except for the piano. And bridge.

As many will tell you, I'm not shy about asking the author questions, and I always rely on the writer to verify my fixes.

Writers usually have something definite to communicate, and a good editor should help them get the message across more effectively and efficiently.

Question: *Since your role is often to "improve" other people's sacred work, the stage seems set for potential adversarial relationships. How do you avoid this situation?*

Cyrus answers: If the author and editor share a common goal: namely, communication with the targeted audience: they shouldn't be adversaries.

In my career I've only run into two or three writers who thought their words were of such divine purity that they shouldn't be sullied by the editorial touch, much less questioned.

My view is that when I read something, my eyes are the customer's, the final reader's, eyes. If I can't figure out what you're saying, then odds are that the customer won't be able to either. Most writers get this sooner or later, and most sooner than later, I'm happy to say.

Of course, we're talking about a *good* editor here, one who practices editing not only as a craft but also as an art. One who knows not only all the grammar and readability stuff but one who also believes almost religiously that language can and should communicate and who has the instinct and skill to make it happen. Maybe I also should add that this good editor I'm talking about picks his or her battles carefully and is not a jerk.

The author and editor should be a team, working to meet that goal I spoke of. Both are professionals and should respect each other's knowledge and skills. If one or both don't, then the game is over. Nothing good will happen. And, bottom line, it's the author's name that goes on a piece, not the editor's.

A good writer/editor relationship develops over time. Like a courting couple, they need to get to know each other. Once there's trust and good give and take, ah, then you're in business. When this happens the author can concentrate on content, which is his or her job, knowing the editor will ask the questions that need to be asked and polish out any weirdnesses that have crept in.

Question: *Technical writing/editing is frequently at the tail end of the process and very frequently ALL of the slack in the*

schedule has been devoured by other processes. Given this unfortunate situation, maybe you can help us improve the process by telling us a bit about the quality of writing you see these days, your biggest pet peeves concerning the process, and what you think the rest of us might do to help you help us.

Cyrus answers: Ah, yes, schedule. Editing is definitely at the low end of the food chain. My work is usually marked by an asterisk on the schedule, the asterisk meaning "and here a miracle happens."

Schedule slips, like deadlines, are facts of life. What will help the process? Professional courtesy for one thing. Don't assume your editor can do the impossible or lives for all-nighters. Do what you can to be fair. Also, keeping your editor informed of what's coming when helps tremendously. And, above all, don't lie to your editor. don't tell him or her what he or she wants to hear or what you *want* to happen. Tell the truth. It will serve you both.

The writing I see these days is pretty good, but all writing needs editing. Mine does. Yours does. If you're writing you're concentrating on content, which is as it should be. But like the old saw about not being able to see the forest for the trees, a writer can't see the clarity for the content. Your editor is trained to see the clarity and help provide it.

And here's the best advice for a writer I've ever heard. If you're getting ready to write something, ask yourself two questions: (1) Who am I talking to? and (2) What am I trying to tell them? Once you've answered these questions very specifically, start writing. But use these two questions as touchstones while you're working. They're your compass points.

About the author. *Jim Terrel is President of Cedar Creek Process Designs, Inc. His interests include project management and dynamic process modeling and execution tools.*

If your editor Can't figure out what you're trying to say, odds are your customer won't be able to either.

Questions & Answers on the CMM/TQM Relationship

by Mark C. Paulk

Question: *I would like to know what you think the relationships are between TQM and its philosophies and the CMM. I have been reading more and more articles lately stating that the philosophies of Deming, Juran, and Crosby are the basis for the CMM. I also am seeing more articles, similar to yours, merging ISO 9000 and the CMM. I do not see the connection to the TQM founders.*

It seems to me a company could be ISO 9000 and CMM level 4 and be creating cement rain coats. They may be the highest quality cement rain coats ever manufactured. They may be manufactured with very mature processes. The company may be following all their process definitions right from the ISO 9000 guidelines. But what good is it if no customers want cement rain coats?

TQM is customer focused. It has as its foundation the needs of society. Deming's point is that healthy companies with healthy employees are necessary for the survival of a society. What he documented with his 14 points and his 7 deadly diseases has little to do with the CMM. Maybe one is about the why and the other is about the how.

The precepts and practices of TQM are essential. The precepts and practices of the CMM are essential. ISO 9000 certification seems to be becoming another essential. These are all complementary. They are also synergistic. They have some common points, like continuous improvement, but that does not mean they have come from the same place or are meant to do the same things. Good, logical, and practical solutions to problems such as what TQM and the CMM are attempting to solve will overlap.

I am not very far along in this journey. I am very likely dead wrong.

What do you think?

Paulk Answers: CMM only attacks the process side of TQM, and specifically for software engineering, although the principles can be applied to any (engineering) discipline.

There are some very important aspects of TQM that are deliberately not addressed in the CMM. What value, for example, would the SEI add to a discussion of the "people" issues? Is there anything that's software specific about people?

The obvious answer is no, yet we can add value in the software process arena, while at the same time we should recognize the importance of the people issues in enabling software process improvement.

As I said, we made a deliberate decision to focus on the area where we could make a major contribution. Things change over time. . .

The desire of some organizations to have a people CMM to help them build their human resource has led to the People CMM. The word software isn't mentioned a whole lot, and it was written by HR-knowledgeable folks.

We are drafting a chapter for CMM v2 on interpersonal skills and organizational change, not as key practices that describe maturity levels, but in the context of enabling process improvement and pointing to better sources of information, such as the P-CMM, Deming, etc. A lot of excellent work has been done by folks much more knowledgeable than we about interpersonal skills!

We also Don't (currently) talk about strategic business planning or customer satisfaction explicitly, which are some of the other important TQM topics. We may do something about those in CMM v2 also, but we are the *Software Engineering Institute*, and we need to stay focused on our primary mission, while not ignoring the critical interdependencies: a difficult balance to maintain.

The CMM was inspired by Crosby's maturity grid, but it has evolved a lot over the past several years. If you'd like to track its evolution, you could read: Mark C. Paulk, "The Evolution of the SEI's Capability Maturity Model for Software," *Software Process: Improvement and Practice*, Vol. 1, Pilot Issue, Spring 1995, pp. 3-15.

Question: *Will it cost as much to produce a product when on level 1/2/3/4 as it will producing the equivalent product when on the level above? Has any company really performed this experiment? A 2 by 2 matrix?*

Paulk Answers: The ones who have data have demonstrated to their satisfaction that it's cheaper and quicker to build products at a higher maturity level.

Question: *Or does the product automatically change when you produce it when you're on a different/higher level?*

Paulk Answers: That's a business decision. You have added opportunities on the cost/functionality tradeoff curve.

Question: *Of course, workers will give the product a higher subjective value and so does management, hence the selling price is likely to be higher.*

Is the cost (as perceived by the manager) to produce a product knowing how the total cost is distributed over all activities involved less than the perceived cost when nothing is known about the cost distribution?

Will you charge the same for a software product when you're on level 2/3/4/5 as you charged for an equivalent product (functionality perceived by the customer) when you were on the level below?

Paulk Answers: That's a business decision.

Question: *If the cost of developing software products really decreases as you move up the CMM ladder, how will you be looked upon by potential customers if you claim to sell/produce products with the same quality as your competitor's notably cheaper products?*

Paulk Answers: Almost by definition, it won't be the same quality and cheaper. It will have to be higher quality and cheaper. You're forgetting rework costs. About 40 percent of software project costs are in fixing bugs to get something shippable. If appraisal/prevention permits you to build stuff for 30 percent less, by definition It's going to be a higher quality product. See the TQM literature.

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

Java Klatch/AFOOT

Joint Meeting Slated

Mark September 9th on your calendars! That's the night Desmond D'Souza of ICON Computing will speak at a special joint meeting of Java Klatch and the Austin Forum for Object-oriented Technology (AFOOT). His topic is "Frameworks and Components with UML-Compliant Models and Java."

The Java Klatch/AFOOT meeting will be held from 6:00 to 7:30 p.m. on the Pickle Research Campus. For more information, contact ICON at (512) 258-8437.

SQI Calendar: Meetings and Conferences of Interest

August 28, 1997

-SPIN monthly meeting

6:30 to 9:30 p.m.

Pickle Research Campus, Austin TX

August 29, 1997

QI Seminar

MM: Version 2

10:30 a.m. to 12:00 p.m.

Pickle Research Campus, Austin TX

September 9, 1997

FOOT/Java Klatch joint meeting

6:00 to 7:30 p.m.

Pickle Research Campus, Austin TX

September 16-October 23, 1997

QI Seminar

Object-Oriented Design and Programming in C++ and Java for C Programmers

10:30 a.m. to 12:30 p.m.

Pickle Research Campus, Austin TX

September 18, 1997

-SPIN monthly meeting

6:30 to 9:30 p.m.

Pickle Research Campus, Austin TX

September 30-October 2, 1997

QI Seminar

Software Project Planning and Tracking

10:30 a.m. to 4:30 p.m.

Pickle Research Campus, Austin TX

October 6-8, 1997

7th International Conference on Software Quality (7ICSQ)

Embassy Suites Hotel, Montgomery AL

October 9, 1997

QI Seminar

Java as an Object-Oriented Programming Language

10:30 a.m. to 4:30 p.m.

Pickle Research Campus, Austin TX

October 16, 1997

-SPIN monthly meeting

6:30 to 9:30 p.m.

Pickle Research Campus, Austin TX

October 23, 1997

QI Seminar

Object-Oriented Concepts and Approaches

10:30 a.m. to 4:30 p.m.

Pickle Research Campus, Austin TX

October 23-24, 1997

QI Seminar

Constructing a Software Requirements Specification (SRS)

10:30 a.m. to 4:30 p.m.

ckle Research Campus, Austin TX

ember 12, 1997

-FOOT meeting

00 to 7:30 p.m.

ckle Research Campus, Austin TX

ember 17-18, 1997

QI Seminar

onstructing a Software Design Specification (SDS)

30 a.m. to 4:30 p.m.

ckle Research Campus, Austin TX

ember 20, 1997

-SPIN monthly meeting

30 to 9:30 p.m.

ckle Research Campus, Austin TX

ember 24-25, 1997

QI Seminar

oftware Risk Management

30 a.m. to 4:30 p.m.

ckle Research Campus, Austin TX

ember 10-12, 1997

QI Seminar

oftware Quality Assurance

30 a.m. to 4:30 p.m.

ckle Research Campus, Austin TX