

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 7, Number 1



Winter/Spring 1999

SQI—Renewed, Refreshed, Reinvigorated

by Gary Cobb, Chair, SQI Advisory Group



Working through the winter, the SQI Advisory Council and its new chair, Gary Cobb, studied how SQI could better foster community, offer practitioner-to-practitioner training and networking among peers, provide affiliation for development of premier quality products and services, and become a dynamic educational enterprise for software professionals.

The result—a reshaping of the institute’s mission, strategies, organization, and tactics. In short, SQI has reinvented itself.

The new vision calls for SQI to become the forum of choice for software professionals in the Austin community. Together, with professionals, SQI will provide the catalyst necessary for Austin to have quality software practitioners to develop best-in-class products.

Revised Mission Statement

SQI is a nonprofit, cost-recovery unit within UT’s College of Engineering. It is a multidisciplinary program drawing on engineering, computer science, and business disciplines to address issues in the creation and consumption of high-quality software.

The institute identifies and communicates information on software quality principles, concepts, and technology through seminars, certificate programs, workshops, conferences, support groups, a web site, and publications. Programs are designed for different organizational levels and for private and public entities of different sizes. SQI may also consider future research and development possibilities.

Today’s Market for SQI Products and Services

The academic world of learning and the business world must be brought closer together. The explosion of contractor relationships and high turnover rates of professional employees require the stability of a community-based professional affiliation. The growing gap between today’s skills and tomorrow’s must be halted and closed.

Yesterday, companies recruited employees straight from universities and provided in-house training for technical professional staff. Today, the need for high-level training for professionals is so great that companies no longer have the resources to handle the task. More and more high-level education for professionals is being outsourced.

Meeting Those Needs

SQI will refocus its core offerings, add new activities and emphasis to its programs, delete products or programs that add only peripherally to the aim of developing high-quality software, and continue to offer practitioner-to-practitioner training and networking activities.

The changes will be evolutionary. SQI will transition away from offering symposia and conferences. Instead, the institute will offer packages of related topics through professional certifications, hands-on and problem-solving workshops, seminars, authoring groups, and books.

Need It, Learn It, Use It! is SQI’s new rallying cry.

SQI’s Core Activity

Professional certifications are the core of SQI activities. The current model of providing practitioners with up-to-date and hands-on competency-based skills that are immediately transferable to the workplace continues to be favored by software professionals. Program development will aim to create many synergies between certificate programs and other activities.

Breadth, Depth, Networking

SQI will continue to offer its professional certifications aimed at breadth of knowledge. In addition to seminars, SQI will add hands-on and best-practice workshops for in-depth, applied know-how. Networking and authoring groups will provide professionals the opportunity to expand their knowledge and skills through face-to-face interaction with peers.

SQI Hands-On Workshops

A new category of SQI training will be hands-on workshops scheduled in three-hour evening sessions over several weeks. The subject matter and style of these workshops will be hands-on, practical, and directly applicable to today’s workplace.

Renewed (continued on p. 2)

What’s Inside

Quality Award Criteria	2
Software Reliability, Part II	3
State Auditor’s Report on SWPM	4
SWPM Graduates	5
A License to Engineer	6
SQI Seminars	9
Process Improvement at TDHS	10
SQI Calendar	11

Renewed (continued from page 1)

Tying Workshops to Certificate Programs

Hands-on workshop and seminar topics will complement certificate program topics. This will ensure the creation of synergies and give professionals the option to receive in-depth instruction in one or more areas that are of particular interest to them or their organizations. Instructional content will address current needs of the community. Format will include readings, lectures, exercises, homework, collaborative software, and trained facilitators and editors.

SQI will train facilitators to lead best-practice workshops. These facilitators will lead discussions, help groups develop potential topics for books, and aid in the capture of the knowledge created during discussions. SQI will also train development editors to lead team-based authoring groups.

Development of Professional Knowledge

Through SQI's programs, members of the development community will be able to raise their level of professional expertise, reduce the time to acquire new knowledge and skills, share common experiences, learn from their peers, and build on the ideas of each other.

Tuesday Nights at the Commons

Plans are underway to consolidate some of SQI's training and professional networking activities on Tuesday evenings at the Commons on UT's Pickle Research Campus. This will serve to couple hands-on competency-based training with ongoing and dynamic involvement among fellow practitioners. The result for program participants will be a more synergistic community of software developers and project managers.

The SQI Advisory Group believes that concentrating some of SQI activities at the Commons on Tuesday nights will allow software professionals to more easily manage their calendars and

participate in events. Besides dinner, Tuesday night activities will also include collaborative software support, training programs, hands-on workshops, best-practices workshops, team-based authoring groups, facilitated discussions, focus groups, and research presentations by UT faculty.

SQI—A Learning Organization

Facilitators will be part of a feedback loop that will help SQI stay apprised of the needs and concerns of the community. This feedback loop will also include inputs from focus group sessions that will identify the ideas and concerns of community members and improve the quality of the institute's offerings.

SQI Book Series

The final component of SQI's push to capture, organize, and distribute the vast knowledge of the Austin software development community is the SQI book series. This series will focus on practical information written by and for practitioners. Team-based authoring groups will help build on the existing knowledge of the Austin community, as well as disseminate practical know-how that exists already.

The SQI Success Story

In the past 6 years, SQI has managed training for more than 6,800 software professionals through seminars, professional certification programs, conferences, and a special Master's degree program for software engineers.

Software Quality Award Criteria Now in Place



The Greater Austin Quality Council (GAQC) now offers a software-specific quality award with criteria developed and piloted under the direction of the Software Quality Institute (SQI). This year's team worked with Jim Nelson, GAQC executive director, to enhance the software pilot that was used last year. While last year's pilot had a separate set of criteria, this year the software criteria are integrated with those of the 1999 Malcolm Baldrige award. Thus, all GAQC award applicants whose businesses depend on software development will be using the new software criteria, in addition to the Malcolm Baldrige criteria.

The 1999 team that made the changes from the software pilot consisted of Gary Cobb, Geree Streun, and Lynn Thurmond.

The 1998 Software Pilot

Last year, a team composed of Gary Cobb of Dell Computer Corporation and Marilyn Payne of IBM developed the software criteria, which they called CMM lite. Herb Krasner and Geree Streun were final reviewers for the 1998 software pilot.

Developing a lite version of SEI's Capability Maturity Model (CMM) version 1.1 was a strategy the committee adopted in developing software-specific quality award criteria. While the CMM has been in public use for several years, it had to be tailored, since a CMM-based assessment typically takes more than a week to perform. The challenge of the CMM-lite approach was to allow a complete examination within a one-day site visit.

The 1999 Changes to the Software Pilot

This year's team decided that a typical Level 1 applicant for the GAQC award does not have adequate software maturity to withstand a Level 2 examination based on the CMM. This decision caused the award criteria to be moved up one level, so that now, a Level 2 applicant for the GAQC award will respond to criteria from Level 2 of the CMM. Likewise, a Level 3 applicant for the GAQC award will respond to criteria from Level 3 of the SEI CMM. Also, the scoring sheet for software was separated from the

scoring sheet of the Malcolm Baldrige. With these changes, the integrated 1999 awards criteria were released at GAQC's Spring Seminar on March 29th.

Benefits to Business

If someone came into your business and said, "I know how to cut your effort and schedule for software developments in half and deliver a quarter the number of defects to your user community," what would you say? As shown in figure 1, going from L1 to L3 beats even those numbers. Those in the software industry who have spent the time and effort to improve their software development processes and reduce the defects of that process, using the CMM as a guide, have found supporting evidence.

One of the main causes for this reduction in effort and schedule is the focus on software processes, team training on new processes, the use of a measurement program to guide the continuous improvement, and the effectiveness of team self-assessments to drive the institutionalization of the

Award (continued on p. 6)

Software Reliability—Keeping the Bugs Out

by Allen Johnson



Part I of this article, which was published in the Fall 1998 edition of Software Quality Matters, discussed where software bugs come from and how to keep them out. The author suggested implementing software reliability engineering (SRE) methods into the development life cycle as a way of meeting the customer's reliability requirements and how testing could be used effectively to remove bugs.

Now in Part II, the conclusion of this article, Johnson discusses improving system reliability, growing reliability, testing reliability growth, implementing fault tolerance, and the characteristics of universal fault tolerance.

Improving System Reliability

To improve the reliability of a system, you must find and remove requirements defects, design defects, process defects, documentation defects, and training defects. The rate at which defects are removed determines the rate of reliability growth. Too often documentation and training are overlooked, and then they end up blowing you out of the water.

The defect removal process should be started in the life cycle and be very aggressive, particularly during the requirements and design phase. Disciplined inspections and proofs can be very efficient at removing defects. Remember: "Pay me now, or pay me much, much more later." Also: "We always have time to do it over when we don't do it right the first time if the project is not canceled." When you do it over, however, you can never do it as well as it could have been done the first time because you always have to live with many of the architectural mistakes you made the first time.

Growing Reliability

Reliability grows by improving the design, improving the effectiveness of testing and inspections, improving the learning and familiarization of both developers and users with the product, and improving management processes and decisions. Most reliability problems can be traced back to poor management. This includes both management before and after the product is shipped.

After the product is shipped, reliability of the system may be largely in the hands of a system administrator and help desk personnel. Systems are becoming more complex for system administrators to deal with, and good system administrators are rare. Based on this, it can be predicted that, from the customer's perspective, the reliability of most commercial systems will get significantly worse in the next few years until something is done about the problem. This problem is exacerbated by the fact that each new release of a product is much more complex than the previous release. Expect the demand for system administrators and help desk personnel to continue to increase rapidly.

The rate at which reliability grows depends on the following three factors.

1. How rapidly defects are discovered
2. How fast corrective action can be identified and implemented
3. How soon the impact of the change shows up

Reliability Growth Testing

Reliability growth testing requires good planning. This involves establishing clear goals and objectives to be accomplished, selecting the procedures to be followed for testing and fixing defects, and providing the configurations to be tested and the resources to perform the testing.

The next step is to perform the testing, identify defects, and gather metrics. Based on the defects identified, corrective action is taken and metrics are monitored to determine when successful completion has been achieved. Once sufficient corrective actions are completed so that reliability goals are achieved, then the testing is successfully terminated.

If it is determined that reliability goals cannot be achieved in the allotted time or even with a reasonable extension, then testing should be terminated. Typically, this is an indication that there are basic design problems that need to be studied, and testing should not be resumed until those design issues have been successfully addressed.

Implementing Software Fault Tolerance

What if you can't afford the consequences of any bugs? Perhaps you cannot afford

for the system to be down for any period of time because the cost may be in the millions of dollars or life depends on the application running correctly. In these situations, it is more cost-effective to make the software fault tolerant.

Actually, it is possible for implementing software fault tolerance to be more cost-effective in less critical situations because of the reduction in testing time and in maintenance time that should result. You still need to go through the process of removing bugs, but now the code itself will help make the process go faster.

The first step is to be able to distinguish between correct and incorrect inputs and operations in the software itself. Once the software can distinguish between correct and incorrect inputs, it must be able to always map a correct input to a correct valid output and map an incorrect input to an invalid output that can be recognized and handled. If the incorrect input is handled in such a way that the system can determine what the correct input should have been and then perform the correct operation and deliver the correct output, then the system is fault tolerant.

There are various techniques for implementing fault tolerance that involve concurrent error detection mechanisms in software. In the distributed multiprocessor system environment we typically have today, fault-tolerant mechanisms implemented must deal with the following types of failures.

- Omission faults
- Timing faults
- Sequential consistency faults
- Byzantine faults

An omission fault causes an element to fail to respond to a trigger event. A timing fault causes an error even though an element responds correctly to a trigger, but the response is not in the allowable window of time, and it arrives either too early or too late. A sequential consistency fault generates an error when an error-free operation is dependent on the correct order of execution or the correct order of response. More general in that it includes the previous three classes, a Byzantine fault is any fault where an element's behavior or response deviates from what is specified.

Bugs (continued on p. 4)

State Auditor's Office Gives High Marks To SQI Software Project Management Program

by Cinda L. Cyrus



SQI's Software Project Management Program (SWPM) does indeed enhance the project management skills of state agency software developers. That was the overall conclusion reached by the Texas State Auditor's Office (SAO) after conducting a recent in-depth audit.

SWPM provides intensive training in several areas, including inspection activities, documented plans, risk management, planning and control, metrics, and estimation. The Software Quality Institute, which offers the program, is in the College of Engineering at The University of Texas at Austin.

The SAO study was a systematic, unbiased, statistically validated measure of the program's effectiveness for employees of state agencies who completed one of the eight 48-week certificate program offerings in the period 1993 to 1997.

Changes in Behavior

Information gathered from program graduates, their managers, and their clients indicates that project management skills improved as a result of the SWPM program. Almost all of the state agency graduates (93 percent) reported that the training caused a change in their method of project management, and all of the managers

interviewed considered the training to be beneficial.

State graduates reported statistically significant changes in four subject areas: inspection activities, documented plans, risk management, and planning and control techniques. The areas of most significant change were performance of inspection activities (inspection of artifacts, code, processes) and use of documented plans.

Before the training, 34 percent of respondents reported use of inspection activities "sometimes" or "most of the time." After the training, 78 percent of respondents reported use of the activities at the same frequency.

The documents that the greatest number of respondents use most frequently are the software requirements statement and the software test plan.

Though the majority of respondents (79 percent) reported that estimation techniques were important or very important to successful completion of software project management, no statistically significant behavior changes were noted in the use of estimation techniques or metric selection and imple-

mentation after training. All respondents, however, reported that they used project progress metrics "some of the time" or "most of the time" after attending training, as opposed to 57 percent before the training.

Applying SWPM Lessons in the Real World

The SAO study also revealed several factors in the workplace that contribute to the successful application of information learned by SWPM graduates.

- Active management support of information resources departments' efforts to implement new procedures
- Providing training to help users understand and better participate in new procedures
- A champion for software project management/quality assurance improvement or a designated individual to spearhead process improvement activities
- Multiple SWPM graduates in an agency lead to better understanding of and support for changes in project management techniques

Bugs (continued from page 3)

Characteristics of Universal Fault Tolerance

In general, a system must have some essential characteristics—including observability, controllability, identifiability, performability, survivability, responsibility, communication, consensus, timeliness, and rationality—in order to achieve universal fault tolerance. If these characteristics are present, then the system can not only be fault tolerant but it can also be secure and safe.

Without a system being dynamic (changing constantly), it will never be completely secure. Without a system being dynamic and having some degree of consciousness, it can never completely adapt to unforeseen events that were not anticipated by the designers. Current fault-tolerant systems available today do not have all these characteristics. Therefore, there is always some limitation on the fault tolerance implemented. It can never be 100 percent.

It's Music to Our Ears, SQI Says



SQI was all smiles when the Texas State Auditor's Office (SAO) released its report on the effectiveness of SQI's Software Project Management certificate program.

The SAO concluded that SQI's program enhances project management skills of state agency software developers, particularly in the performance of inspection activities and in the use of documented plans.

"It was music to our ears," said Joanne Click, SQI program director, speaking of the report. "We are very proud of the highly

favorable marks received by our flagship program," Click said.

This report validates internal surveys conducted by SQI among the 200 SWPM graduates over the past five years and generally substantiates informal comments received about the program from the Austin software development community, Click said.

The SAO study marks the first time that the agency has undertaken a survey of training provided to state agencies across the board. The SAO plans a follow-up report in several years to monitor process improvements over time.

Conclusion

In order to achieve a reliable software design, a disciplined design process minimizes the number of defects that get put into software. Reliability growth testing and defect tracking enable an organization to reach a desired level of reliability by finding and fixing defects fast. Customer

expectations for reliability must be clearly understood and met.

When the cost and/or consequences of failures are great, some form of implementation of fault tolerance should be considered.

Note: for references, please contact Johnson at allen@rasgroup.com

37 Students Begin Quest for SWPM Certificates

SWPM Thirty-seven students and 12 instructors launched Sequence 10 of SQI's Software Project Management (SWPM) program on March 16. The students will graduate from the 48-week certificate program in May 2000.

The SWPM curriculum consolidates bodies of knowledge from the Project Management Institute, the American Society for Quality, and SQI. Top-flight software practitioners from the community work together as teams in the program to assure that the students achieve the professional competencies required in their project management careers.

Students in the new class represent the following companies: Addressing Your Needs, Alcatel USA, BMC Software, Cadence Design System, Catapult Systems Corp., City of Austin, CSC Financial Services Group, Dell Computer, EDP Contract Services, IXC Communications, Micro Information Products, National Instruments, Office of the Texas Attorney General, On-Board Software, Page One/Proximity, PSW Technologies, SBC Technology Resources, Tivoli Systems, TX Dept. of Health, TX General Land Office, TX Natural Resources Conservation Commission, USAA, U.S. Air Force, and UT Applied Research Labs.

Leaders and Instructors

SWPM leaders and instructors represent the best and brightest of software professionals.

Bob Futrell of Motorola and Marilyn Robertson of SQI keep the complex SWPM program running smoothly. As SWPM Instructor-in-Charge, Futrell orchestrates the teaching teams and curriculum. Robertson, who is SQI's incredibly competent SWPM coordinator, manages the curriculum, classroom, and electronic resources that support the program.

Instructors, mentors, and project masters include Gary Cobb (Dell), Dennis Frailey (Raytheon), Bob Futrell (Motorola), Richard Grant (Clinical Psychologist), Laura Grosvenor (UT-ARL), Allen Johnson (Rainbow Analysis Systems Group), Herb Krasner (Krasner Consulting), Cecil Martin (Martin Management Service), Mike Murray, Jack Odom (Cadence Design Systems), Mike Pore (Sematech), Don Shafer (Athens Group), Gerec Streun, and Gail Taylor-Russell (Russell & Russell). The mentor for Sequence 10 is Linda Shafer (Motorola) assisted by Diane Scheffelin

(Texas Department of Human Services) and Jack Odom (Cadence Design Systems).

SWPM Passes Review

A curriculum review team composed of SWPM graduates reviewed the program this winter and recommended changes to assure the program continues to meet the needs of Austin's software companies. Each member of the review team dedicated 1,000 hours to the intensive review.

SQI thanks each of the reviewers for

their contribution to SWPM. The reviewers were—

- Bob Futrell, Motorola, review leader
- Cheri Andrews, EDS
- Celeste Armstrong, Motorola
- Robert Culbertson, Cisco Systems
- Lisa Giguere, Texas Land Office
- Paul Patterson, Intelligent Technologies
- Diane Scheffelin, Texas Department of Human Services
- Ann Weiss, Wyatt Consulting

SQI Salutes Recent Graduates Of SW Project Management Program

Congratulations, December graduates of SQI's Software Project Management program! We salute your dedication, professionalism, and hard work during each of the 48 weeks of the course!

With your successful completion of this comprehensive certificate program, you take back to your organizations—large and small—a body of knowledge encompassing the latest and most immediately applicable methods and techniques for managing software projects effectively and efficiently.

Nearly 200 software project managers have now graduated from the Software Project Management program. The competency-based certificate program has graded

homework assignments and a series of four exams. Classes, which include lectures and class exercises, meet once a week for 48 weeks and are taught by project managers and experts in the field—all practitioners themselves. The curriculum covers all areas addressed by the PMI and CSQE exams.

Additional Information on SWPM

For additional information on the SWPM certificate program, contact—

Marilyn Robertson

Phone: (512) 475-8649

E-mail: marilyn@sqi.utexas.edu

You may also visit our web site at www.sqi.utexas.edu

Congratulations to SWPM Students!

Patti Abkowitz, 3M

Cheri Andrews, EDS

Maurice Barnett, Dell Computers

Andrew Bergstrom, Pencom Web Works

Bob Black, Cisco

Audrey Carroll, TX Comptroller of Public Accounts

David Chambers, Xetel Corp.

Chris Elder, Texas Guaranteed Student Loan

Stacey Fox, Dell Computer

Lisa Giguere, TX General Land Office

Andy Gillam, Cornerstone Integrated Services

Jonathon Green, 3M

Ron Greger, TX Comptroller of Public Accounts

Mike Haney, The Athens Group

Corey Horton, Sterling Information Group, Inc.

Mohamed Kinj, Dell Computer

Pamela Kopfer, TX Dept. of Information Resources

Margo Meyer

Tava Michalik, Texas Dept. of Human Services

Suresh Nair, VHA

James Nold, Dell Computer

Joe Pacheco, Randolph Air Force Base

Mark Ragan, Switch Solutions

Peter Reitmeyer, IBM

Regina Rousseau, TX Dept. of Information Resources

Michael Schucking, Pryor Systems

Tom Sethre, Sterling Information Group, Inc.

Dean Thomas, Catapult Systems

Dianne Thompson, Comptroller of Public Accounts

Amy Vanderburg, TX Dept. of Human Services

Ann Weiss, Wyatt Consulting

James Wiatrek, Catapult Systems

Mary Williams, TX Dept. of Human Services

Software Engineering Services in Texas— Shedding Light on the Licensing Issues

by Herb Krasner



Licensing has finally come to software engineering in Texas. Now the questions arise.

- WHO can call himself or herself a software engineer?
- WHAT companies can legally offer software engineering services?

The following recent letters offer a clue to the answers. The name of the company and individual involved have been changed to protect confidentiality.

To: Executive Director, Texas Board of Professional Engineers

My company, XYZ, is exploring the possibility of expanding into the Texas marketplace. To that end, I am tasked with determining the ramifications of using the term *software engineering*.

Can you provide a clear definition of what the inappropriate use of the term would be? In a consulting firm, are all the engineers required to have a PE license for the firm to use the term *software engineering* in its advertising?

**Sincerely,
Mr. _____
XYZ Corporation**

Dear Mr. _____,
For over 60 years, the State of Texas has regulated the practice of engineering. This is done under the Texas Engineering Practice Act and the associated rules of practice promulgated by the Texas Board of Professional Engineers. The regulation is accomplished by licensing qualified engineers and restricting public practice and representations to those who are licensed. Violations of law are criminal offenses, but the Board more commonly uses judicial injunctions and administrative penalties (fines) to enforce the Act.

Only about one in five engineers has a license. The license enables engineers to legally represent themselves as *engineers* to someone outside the confines of their full-time employer; offer to perform, or perform, engineering services for someone other than their full-time employer; and to design and/or implement engineering projects for a public work. Any company that offers engineering services for sale must have a licensed engineer who is a regular, full-time employee on staff. That licensed engineer must personally perform or directly supervise the engineering work.

It appears that your company's offer of software engineering services would require

your company to have a licensed engineer on staff to meet the requirements of the Act. If you do not currently have such an employee but feel that you have some who are qualified, you can download a pdf application packet from our web site at www.main.org/peboard/

Right now, the turnaround for those already licensed in other states is about four weeks; the turnaround for new applications varies depending upon the situation.

Licensing for software engineers may be a little more involved than with more traditional disciplines of engineering. Texas licenses senior software engineers based on an experience record sufficient to justify an exam waiver. Those applicants who have a PE license in other states but who wish to be known as software engineers may use that title at their discretion upon receipt of a Texas PE license.

There are several articles on our home page (including one on software engineering) that may better elaborate on our Board and the law.

**Sincerely,
Executive Director
Texas Board of
Professional Engineers**

Award (continued from p. 2)

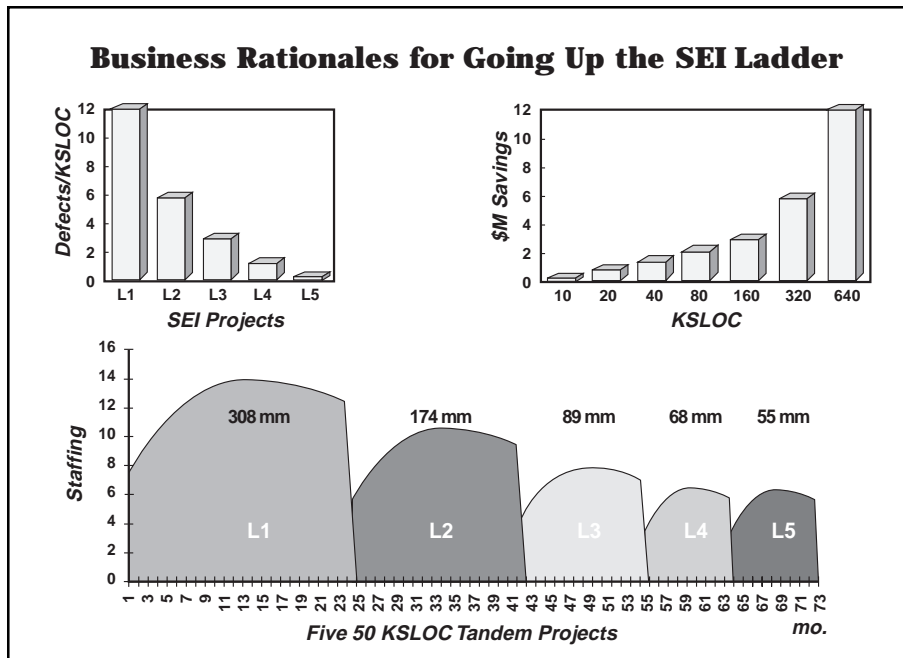


Figure 1. Benefits of Climbing the SEI Ladder

process improvement. Also, driving our software defects early in the life cycle is a key contributor to the reduction of effort and schedule, as shown in figure 2.

From 1999 on, software-developing companies in the greater Austin area will have a new tool for assessing their quality and productivity—the GAQC's awards criteria—thanks to the efforts of these SQI teams.

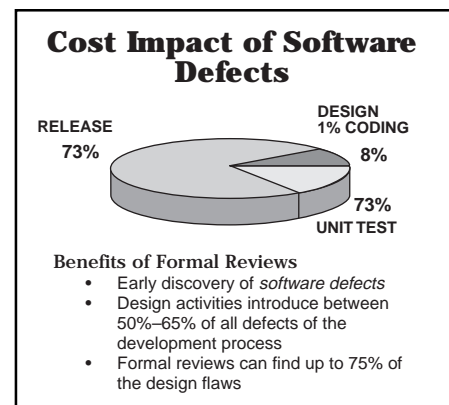


Figure 2. The Cost of Software Defects

A License to Engineer High-Quality SW Systems

by Herb Krasner



In a recent issue of *IEEE Software*, John Charles attempted to characterize the initiative in Texas to license software engineers in a column entitled *A License to Code*.

I am assuming that Charles intended to evoke a reaction from the readership with that particular title, instead of something like A License to Engineer High-Quality Software Systems. Readers of *IEEE Software* know that the licensing issue is not about coding.

The State legislature says it wants to protect the public from incompetent individuals claiming to be able to engineer software systems. Is that the right objective? If so, what does it mean? What many of you might not know is that in the State of Texas, the term *engineering* is a *reserved* word, and you can get sued for using it inappropriately when advertising your services to the public.

Problems with the Texas Approach

There are a few key problems with the current approach in Texas.

- A lack of vision and understanding of what a professional software engineer (PSWE) would be
- A lack of agreement on the bodies of knowledge necessary for an apprentice SWE and a PSWE
- Not addressing the essential issues of what a SWE is and why software engineering is different from other engineering disciplines
- A lack of agreement on the interdisciplinary educational requirements for producing a capable apprentice SWE
- Ignoring 99.99 percent of the SWE practitioners in the state

I believe that the Texas Board of Professional Engineers' attempt to license professional software engineers in the State of Texas is guilty of low expectations. Charles's article might lead you to believe that this licensing movement potentially applies to all software developers in the State of Texas. Not true. It is intentionally limited to software developers performing or offering to perform engineering works for the public, as strictly defined by the Texas Board of Professional Engineers.

The board's concern is to prohibit a non-engineer from selling engineering services publicly to develop or modify software that could affect the health, safety, or welfare of the public. This does not affect the internal use of titles of persons or organizations within a company. It does affect how those persons or their services are represented outside the company—that is, to the public. For example, you may not tell a potential client that a software engineer will perform a task unless that individual is a licensed software engineer.

In other words, it is really only attempting to recharacterize the work that current professional engineers are doing under another engineering discipline. It is not attempting to bring other SW development efforts underneath an engineering umbrella, which is too bad because that is what is really needed. For example, all of the Y2K (and similar) problems with information systems across the state would not be addressed because they do not fall under the category of engineering works. Yet a cause of Y2K problems is the lack of competence exhibited by those who originally wrote the software. In other words, the Texas initiative will ignore most of the problematic practice in the State.

Defining a Body of Knowledge

The licensing and certification of SWEs must flow from an agreed-upon and defined body of knowledge in two stages: (1) for an apprentice SWE and (2) for a PSWE.

Stage 1 represents the knowledge necessary to pass a fundamentals exam. Stage 2 represents the knowledge necessary to pass the Principles of Practice Exam.

In either case, the knowledge needed is multi-disciplinary, consisting of topics from computer science, computer engineering, systems engineering, engineering management, math, social and cognitive sciences, other sciences, business, and the application domain. This leads us into accreditation issues and, thus, into big-time academic politics at Texas universities.

An Academic Definition

The definition of software engineer provided in Charles's article is a good example of what happens when you take an academic perspective in defining software engineering: you get a laundry list of existing course titles from the computer science department catalogue,

plus a smattering of other stuff. Although this makes accreditation of existing departments much easier, it doesn't solve the essential problem of stamping out poor-quality software. It also reinforces the roles of the traditional academic disciplines, instead of what is needed.

Charles's definition is flawed and must be fixed in order to align with the emerging body of knowledge. It should exhibit characteristics that purport to combine the disciplines of computer science, computer engineering, computational systems engineering, and application domain engineering. Charles is missing these key words from his definition: *disciplined methods, learning from experiences, quantifiable approaches, multi-disciplinary, and objective-oriented behavior*.

An Alternative Definition—Let the Debate Commence

With thanks to the many pioneers of the emerging field of SWE, here is my alternative definition of a PSWE. Notice that it starts with a vision or goal-oriented view of what a PSWE should be.

A PSWE is an experienced practitioner who uses sound engineering principles, good judgment, and knowledge of the application domain to create working software systems that achieve given user objectives within stated economic, time, resource, and other constraints. A PSWE uses systematic,

License (continued on p. 8)

Software Engineer Licensing Approved

The Texas Board of Professional Engineers now officially recognizes software engineering as a legitimate engineering discipline and has begun licensing professional engineers in that area.

A complete position statement from the Texas Board can be found at www.main.org/peboard/softweng.htm.

The Texas Board is now licensing software engineers through a waiver process, until examinations are in place, probably in one year.

For additional information, please contact Don Bagert, associate professor of computer science, Texas Tech University, at bagert@ttu.edu

License (continued from p. 7)

disciplined, and quantifiable approaches for the definition, development, operation, and evolution of computer software.

Software Engineering Tasks

While the essential aspects of software engineering would best be described by performing a task analysis that links back to the previous empirical studies of SW development undertaken by my team at MCC in the mid 1980s, perhaps we can all at least agree that the following are the core tasks of an individual SWE (adapted from Parnas, 1998).

1. Analyze the intended application to determine and document the requirements to be satisfied
2. Participate in the design of the computing system, determining which functions will be implemented in hardware or software and selecting the basic components of the architecture
3. Analyze the performance of a proposed design to make sure that the requirements can be met
4. Design the software (including module identification, module interfaces, the structure of individual programs and databases) and documenting all design decisions
5. Analyze the software structure for completeness, consistency, and suitability for the intended application
6. Implement the software as a set of well-structured and well-documented programs and databases
7. Integrate new software with existing and/or off-the-shelf software
8. Perform systematic and statistical testing of the software and integrated system
9. Deliver the software and system to customers in an acceptable condition
10. Revise, enhance, and evolve software systems so as to maintain their conceptual integrity, while keeping documents complete and accurate
11. Continually improve the effectiveness of this essential process (tasks 1 through 10) by systematic learning from experience, mentoring, and other sources

Where computer science fits into the this description is obvious to some and not obvious to others. Of course, individual SWEs usually work in teams on identified projects with specific objectives, both technical and business. And here is where consensus begins to break down on what is and what is not software engineering.

Parnas, for example, specifically excludes project management from software engineering because he claims it to be broader in subject matter, but most software projects fail because of project management failures (objectives not achieved). Other experts include other supporting areas and related specialties (e.g., human factors, SQA, CASE tools, etc.). Watts Humphrey includes step 11 above as I have, others don't.

So, then, with much subject matter yet to be debated, the controlling issue comes down to who will define the body of knowledge and how it will be used in the licensing process. A strawman software engineering body of knowledge now exists from the joint task force of the IEEE CS and ACM and should nicely fuel the debate.

I encourage all *IEEE Software* readers to carefully examine the strawman; it has

a disciplined manner in industrial settings, and many require extensive additional education in order to reach a level of basic professionalism. Most of the companies they go to don't even realize that they are receiving an apprentice SWE with limited skills. The job descriptions of those companies exhibit a naive view of what is required; if these companies think that software engineering is having knowledge of certain languages and operating systems, they have been duped.

How Do We Grow a PSWE?

Several paths are needed. Because the academic backgrounds of those entering the software engineering profession are currently diverse (CS, EE, Math, MIS, etc.), several apprenticeship approaches will be needed. A foundations examination, based

“ Send me your comments. Let's get a dialogue going! ”

strengths and weaknesses. Other voices in the debate include those representing the SWE Code of Ethics, ASQ's Certified SW Quality Engineer, and SQI's Software Project Management Certificate Program.

Opportunities for Growth

I hope the Texas initiative encourages the growth of several recognized specialties in software engineering. At the top of my list is the senior software-systems architect, who I think should be a PSWE. This is the individual who takes full responsibility for the success of the systems that he or she oversees. This was positively reinforced by our observations of the most successful large software projects from our MCC field studies (Curtis, Krasner, and Iscoe, 1988) in which such a person effectively guided the successful project team from an integrated technical perspective.

I would like for the Texas initiative to be successful in terms of enabling our workforce's capability to routinely produce quality software within the given constraints, but I have some doubts. Here in Austin, for example, teaching the foundations of software engineering has long been clouded by an annoying turf war. Why the various departments at UT (e.g., CS, ECE, MSIS) can't cooperate to create a multidisciplinary software engineering degree program is beyond my limited ability to understand.

What I do know is that the graduates of these disjointed programs are not necessarily producing high-quality software in

on a fundamental body of knowledge, can be used to establish basic credentials that derive from formal educational requirements, as well as to identify the weaknesses to be addressed through apprenticeship. The Principles of Practice Exam ensures that the PSWE has reached the level of professionalism implied by the preceding definition.

Why is SW Different from Other Engineering Disciplines?

Most modern systems are not just software. The scientific foundations of software engineering, however, are different from those of physical systems engineering due to the largely conceptual nature of the tasks, and, therefore, the technology underpinnings tend to be unique. Software is not generally constrained by physical limits (other than computer resource limits). Thus, the range of possibilities is harder to determine, and the outcomes are far less certain. Cost of ownership is primarily in design (rather than manufacturing), thus affecting the nature of the quality processes used and how evolution occurs.

What's Your Opinion?

What do you think? Have I struck a chord? Rung a bell? Irritated you? Let's get some dialogue going! Send your comments to me at hkrasner@cs.utexas.edu or to newsletter editor Cinda Cyrus at cyrus@sqi.utexas.edu

Register Now for SQI Spring & Summer Seminars



SQI seminars offer software professionals the opportunity to develop new skills and enhance current ones. Ranging in length from one to five days, each course is designed to provide real knowledge immediately applicable in the workplace. The following courses are currently on the schedule.

Object-Oriented Development Using the Unified Modeling Language

Charles Richter, instructor

This five-day seminar, offered May 10, 13, 17, 24, and 27, leads attendees through a complete OO analysis and design method based on the UML. The course begins with an introduction to basic OO concepts and design principles and basic concepts and notations for static and dynamic modeling and design. These topics are followed by discussions of more advanced concepts and UML mechanisms for extending the notation, design heuristics, system architecture, class design, and frameworks and design patterns. No knowledge of OO concepts or of OO development is assumed.

Object-Oriented Programming in Java

Charles Richter, instructor

This five-day course, offered May 11, 14, 18, 25, and 28, covers the fundamentals of programming in Java. Both the language and some of the standard packages are covered, and numerous exercises and laboratories will provide hands-on experience. The course is designed for those wanting to write applications or applets in Java.

Advanced Object-Oriented Design: Patterns and Frameworks

Charles Richter, instructor

This three-day course, which will be offered May 12, 19, and 26, explains the development, application, and documentation of design patterns and frameworks. The concepts of patterns and frameworks are explained through examples. Because the successful reuse of patterns and frameworks requires good documentation, this course also focuses on how they can be described. A significant portion of the class involves lab exercises during which students will apply and develop patterns and frameworks. Attendee should possess a thorough understanding of basic object-oriented concepts and should have some experience with object-oriented design or programming.

Object-Oriented Concepts

Charles Richter, instructor

This one-day course, which will be offered May 20, introduces basic object-oriented concepts and approaches. The course explores basic object technology and cites its advantages. It also outlines the steps involved in object-oriented analysis and design through a simple example.

Defining and Implementing a Quality Software Process

Cecil Martin, instructor

This two-day course, which will be offered June 2-3, trains software process engineers in the skills required to analyze, design, implement and continuously improve software process systems. Companies have to integrate and harmonize their automated production and non-production process systems, rather than just one or the other. Therefore, a quality software process system must be used to meet the necessary quality, cost, functionality, and time to market demands to stay competitive.

Object-Oriented Design and Programming in C++

Glenn Downing, instructor

This seminar meets on Tuesday and Thursday mornings for five weeks: June 1, 3, 8, 10, 15, 17, 22, 24, 29, July 1, 6, 8. It will cover the fundamentals of object-oriented design and programming with various examples. A working knowledge of C arrays and pointers is essential.

Software Quality Assurance

Cecil Martin, instructor

This three-day course, which is offered July 6-8, focuses on what constitutes quality in software and on the benefits of quality assurance. This course will teach developers

and managers how to plan and document software quality activities and will examine the importance of commitment and measurement process and tools. Verifying the implementation of software quality activities will also be discussed.

Software Configuration Management

Cecil Martin, instructor

This three-day seminar, which is offered August 9-11, is designed to help an organization accurately implement a framework for information systems centered around baselining and controlling automated systems that are critical to business missions. The course will provide both theory and one hands-on exercise to teach the fundamentals and process for configuration management.

Multimedia System Design

Gary Cobb, instructor

This seminar will meet on August 24, 31, September 7, 14 and cover a wide variety of multimedia applications. Authoring tools will be demonstrated and issues surrounding the integration of multimedia content in a distributed computing environment will be discussed.

SQI Seminar Information

Registration fees cover the cost of course materials and light refreshments. Early registration is encouraged; enrollment is limited. All seminars are held at UT's Pickle Research Campus.

To learn more about these courses or other SQI programs, contact—

Software Quality Institute

Phone: (512) 475-6779 or

1-800-687-8012

Fax: 512-471-4824

E-mail: elms@sqi.utexas.edu

We're All Ears!

SQI Calls for Your Seminar and Workshop Ideas



Speak up! SQI is expanding its inventory of training seminars and workshops for software practitioners. We're looking for topics you'd like to see SQI offer and for teachers who are industry practitioners with subject-matter expertise and experience in classroom instruction.

Have a suggestion or an idea? Send it in. We're counting on you!

You can suggest a topic or apply to teach by completing the SQI proposal form on our web site at www.utexas.edu/ftp/coe/sqi/seminars/proposal.html. All proposals will be reviewed by the SQI Advisory Group.

The audience for SQI seminars and workshops includes members of upper management who have responsibility for software projects, software project managers, and individual software contributors.

Process Improvement Success Story at TDHS

by Geree Streun



The Austin SPIN Board instituted an innovative step forward for the organization for 1998 by setting the program focus for the year to the people side of process improvement, with every session enhancing that focus.

A program presented by Betty Flores and Diane Scheffelin focused on the real-life experiences of a major software development organization here in Austin. The program looked at the slow perilous journey to changing the behavior/culture of that organization and its great success!

The Journey Begins

Through a series of stories, Flores and Scheffelin took turns painting a vivid picture of their individual and organizational efforts to bring about process improvement. Flores began by describing the organization, the Texas Department of Human Services (TDHS), which serves more than 2 million active clients and administers more than \$20 million in human services benefits a day.

The organization consists of 200 full-time developers and managers, some of whom have been with the organization 20 to 30 years; there are also approximately 100 contractors. This diverse group supports about 10 million lines of legacy COBOL code. For many years the practiced development process was based on the cowboy legends where heroes get the job done! Over 2 million Texans can thank their lucky stars for those heroes!

The story continued with TDHS's Deputy Commissioner vision that there had to be a better way to develop software, because, in spite of the heroes, there were some issues. Intermediate customers complained about product delivery, TDHS had a lack of credibility with those customers, everyone at TDHS was working huge amounts of overtime, and the turnover rate was very high because of the burnout effect.

SQI's Programs Play a Part

While these issues were coming to a head, SQI's Software Project Management (SWPM) Certificate Program was advertised. The Deputy Commissioner was so motivated that he scheduled 13 of his managers to attend this program. With 13 of the managers attending SWPM classes, he had a cadre in place to attempt to effect a grassroots movement within his organization. Both Flores and Scheffelin are SWPM graduates.

Once the managers were trained, they evaluated their process and found that the defined methodology, tailored from Navigator, was too detailed, cumbersome, and bureaucratic to really facilitate the type of development that TDHS was doing. Even though absolute adherence to that process was mandated for the organization, it didn't take the developers long to learn that they didn't have enough time to follow all those steps.

The key was not to just give up and go with the flow, but to find a way.

The next step was easy for the developers: they just gave the methodology lip service and went on as before! The key to TDHS's effort was not to just give up and go with the flow, but to find a way.

Facing Resistance

Scheffelin then stepped forward to discuss the actual steps needed to make the changes happen. She described how difficult it is to be a new manager trying to sell new ideas to more than 20 managers who have 20 to 30 years of experience at TDHS. Part of the difficulty stemmed from the fact that the old-timers at TDHS spoke their own language, which was sprinkled with many TDHSisms. If you didn't speak their language, why you just weren't in.

The key strategy in her role as process improvement advocate was a multi-pronged attack. It focused on improving the methodology, its method of delivery to the developers, its method of delivery to the managers, and its method of delivery to the customers. The methodology was modified so it only described what had to be done to deliver an on-time quality product, not how it was to be done. How the process was implemented for each project was documented in each project's management plan. Even with this streamlined methodology, there was still resistance because the people issues had not yet been dealt with.

Addressing the People Issues

Scheffelin said she was perplexed about how to cope with this issue. Then she attended an A-SPIN meeting about blending the People-CMM with the Software-CMM to really get process improvement institutionalized.

She studied the first level of the People-CMM and saw that the focus was on communication and training. Training about the methodology that focused on the customers, the managers, and the developers was produced. Then monthly manager meetings that were a mixture of communication and training with the managers were instituted.

Each manager was asked to do a self-assessment of his or her project and to choose one best practice to work on during the coming month to discuss at the sub-

TDHS (continued on p.11)

Software Quality Matters

Editor: Cinda L. Cyrus

© 1999 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

Candy Walser Berry	Gale Hathcock
Debbie Caples	Allen Johnson
Joanne Click	Herb Krasner
Gary Cobb	Mark Paulk
Cinda Cyrus	Marilyn Robertson
Carolynn	Elaine Stanfield
Elmshaeuser	Geree Streun

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
<http://www.sqi.utexas.edu>

TDHS (continued from p.10)

sequent meeting. Then each manager was asked to do a postproject review for every project. The focus was on follow-up and incorporating best practices into the next project and/or correcting the process execution for the next project.

Learning to Communicate

While these were all significant steps forward, communication was still a problem. Scheffelin said that then she and Flores heard another A-SPIN program that offered a solution. The program was about the Game of Questions, which is a straightforward communication technique. They acquired training for TDHS and found that communication was the key for getting past the resistance.

Now the previously resistant managers have established special interest group (SIG) workshops. These workshops are used by the managers to discuss topics like user scenarios, developing text plans, customer sign-off, and postproject reviews. Current project information is used, so real project work is actually completed during these workshop meetings. Having productive meetings when there isn't enough time or staff is really an innovative idea that has been well received by everyone at TDHS.

Change is the Byword

Flores then outlined TDHS's major accomplishments. Change has become the byword for everyone at TDHS, and enthusiasm for change is high among both the developers and managers. Employees are actually pushing for process changes. The customers are excited about receiving requirements specifications and project management plans. In fact, they have become contributors to the projects and are even encouraging additional changes to the process.

Conclusion

Flores said that she and Scheffelin were glad to learn that they could take what they had learned at A-SPIN and apply those techniques to improving their organization. The improvements at TDHS show that the success of developing quality software is dependent on both the process and the people.

As a member of the audience, I was pleased to hear this success story. It proved that in spite of the fact that there are no Silver Bullets, process improvements can and will be made when the organization communicates and works together!

Can Level 2 Key Practices Be Waived for Small Projects?

by Mark Paulk

Question: *A major issue that continues to prevent us from reaching Level 2 is the degree of flexibility we may or may not allow for our very small projects. Small projects in our organization can be those that last one or two weeks and only have one or two people working on them.*

Here's my question. For very small projects, how much of the Level 2 key practices within a key process area (KPA) can be waived? Taking the question a step farther, can any KPAs, such as SQA, be waived for our very small projects? We feel that it would be almost impossible to have our small projects deliver a full project plan, risk analysis, SQA plan, SCM plan, test plans for all levels of testing, and other similar activities normally provided by the larger projects.

Paulk Answers: Rather than saying waived, I'd say have radically different implementations. See the first CMM newsletter on small projects. Certainly any key practice is subject to radically different implementation in very small projects, but the general concerns of each of the KPAs still need to be addressed, albeit at a different level. Look at Watts Humphrey's book *A Discipline for Software Engineering*. It applies the concepts at the level of the individual professional for some thoughts on what the CMM ideas mean in the microproject environment.

Question: *Our objective, of course, is to fulfill the intent of a CMM Level 2 organization. As a possible solution, if we were*

to outline minimized guidelines for our small projects that meet the intent of the Level 2 KPAs, and all small projects were required to follow those guidelines, would an SEI assessment accept such an approach as satisfying the intent of Level 2? By minimized guidelines, for example, instead of a full project plan (using MS Project for example), would a simple 10-item task list suffice? Could formal estimating be waived or replaced by a much simpler hours estimate provided by a programmer?

Paulk Answers: Sounds pretty reasonable to me. The intent is to have a process that is documented, consistently implemented, and a foundation for improvement. You're the best judge of what a reasonable process in your environment is.

Additional Information

For a closer look at CMM questions and answers, see www.sei.cmu.edu/activities/cmm/docs/q_and_a.4.html

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

Practical Software Quality Techniques Conference Held North & South in 1999

It's North versus South again, but this time around both sides win!

After three successful conference years, several Texas groups expressed interest in holding the Practical Software Quality Techniques (PSQT) conference in their own back yard. The result? PSQT '99 South will be held in San Antonio on June 7-11, and the regular PSQT '99 will be held in St. Paul, MN, on October 4-7.

Conference organizers say this experience-based conference offers presentations on practical approaches to software quality.

The following program is planned for PSQT '99 South in San Antonio.

June 7

The following presentations are planned for Monday, June 7.

- *Software Test Automation Techniques: A Disciplined Approach*, Jamie Mitchell
- *A Practical Approach to the Capability Maturity Model for Software*, Version 1.1, William J. Deibler II, Robert C. Bamford

PSQT (continued on p.12)

SQI Calendar—Meetings and Conferences of Interest

May 10, 13, 17, 24, 27, 1999

SQI Seminar
Object-oriented Development Using the Unified Modeling Language
Pickle Research Campus, Austin TX

May 11, 14, 18, 25, 28, 1999

SQI Seminar
Object-oriented Programming in Java
Pickle Research Campus, Austin TX

May 12, 19, 26, 1999

SQI Seminar
Advanced Object-oriented Design: Patterns and Frameworks
Pickle Research Campus, Austin TX

May 20, 1999

SQI Seminar
Object-oriented Concepts
Pickle Research Campus, Austin TX

June 1, 3, 8, 10, 15, 17, 29,

July 1, 6, 8, 1999
SQI Seminar
Object-Oriented Design and Programming in C++ for C Programmers
Pickle Research Campus, Austin TX

June 2-4, 1999

SQI Seminar
Defining and Implementing a Quality Software Process
Pickle Research Campus, Austin TX

June 7-11, 1999

Practical Software Quality Techniques Conference—South
San Antonio, TX

July 6-8, 1999

SQI Seminar
Software Quality Assurance
Pickle Research Campus, Austin TX

August 9-11, 1999

SQI Seminar
Software Configuration Management
Pickle Research Campus, Austin TX

August 23-24, 1999

SQI Seminar
Multimedia System Design
Pickle Research Campus, Austin TX

- *A Practical Application of ISO 9001 to Software Development*, William J. Deibler II, Robert C. Bamford
- *Exploring the Secrets of Successful User Acceptance Testing*, Randall W. Rice, CQA, CSTE
- *Improving Software Quality with Reviews and Inspections*, Karl E. Wiegers

June 11

The following presentations are planned for Friday, June 11.

- *Exploring Software Requirements*, Magdy Hanna
- *Planning and Controlling Software Projects*, David Jones
- *Testing Object Oriented Systems: Test Case Design and Implementation*, Robert Binder
- *Testing Internet and Web Applications*, Randall W. Rice, CQA, CSTE
- *A Systematic Approach to Identifying and Solving Project Problems*, Johanna Rothman
- *Understanding Software Test Automation*, Michael Sowersna Rothman

Additional Information

More information concerning the PSQT '99 South to be held in San Antonio may be found at www.softdim.com

PSQT (continued from p. 11)

- *Introduction to Writing Testable Requirements*, Richard Bender
- *Becoming an Effective Test Manager*, Randall W. Rice, CQA, CSTE
- *Creating a Software Engineering Culture*, Karl E. Wiegers
- *Effective Methods and Techniques for Defining, Analyzing, and Reporting Software Measurements*, Scott Goldfarb

June 8 and 9

Along with 36 concurrent track presentations by software quality practitioners, Tuesday and Wednesday will feature the following speakers.

- Watts Humphrey—*Disciplined Software Teams*

- Robert Glass—*Troubled Software Projects: War Stories and Lessons Learned*
- James Bach—*Good Enough Software Testing*
- Karl Wiegers—*Read My Lips: No New Models*

June 10

The following presentations are planned for Thursday, June 10.

- *Software Test Planning and Design: The Essentials*, Magdy Hanna
- *Developing and Implementing Software Test Plans*, Johanna Rothman
- *Software Project Risk Management: A Quick-Start Workshop*, Joyce Statz



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