ABSTRACT

This paper will present a framework developed by IBM for classifying and analyzing defect data collected during software development. The paper will describe Orthogonal Defect Classification (ODC) and illustrate how ODC can be used to measure development progress with respect to product quality and identify process problems. Next, the paper will present the results of a feasibility study conducted by the Motorola Corporate Software Center, Software Solutions Lab (CSC/SSL) and the Cellular Infrastructure Group, GSM Product Division's Base Station Systems (GSMBSS) software development group using ODC. Finally, GSMBSS's future plans for deploying ODC are discussed.

Introduction

Formal inspections are currently used throughout Motorola to identify software errors near the point of insertion. In 1996 nearly 85% of all product software developers within Motorola reported that they used formal inspections on their development projects. GSMBSS has been conducting Fagan Inspections since 1992. Since then they have amassed a large amount of data on software defects. While analysis of these data has resulted in improved containment, continuing to reduce defects by 10 every two years is becoming increasingly more difficult. GSMBSS, like most software development organizations, has struggled with how to better utilize data collected from defects to measure the progress of the product quality during development, and how to use the data to identify and eliminate process problems.

Traditionally, two distinct techniques, Statistical Defect Modeling and Causal Analysis, have been used to analyze data collected from software defects. Statistical Defect Modeling considers each defect as a random event and attempts to fit a statistical model to the collection of defects. This includes statistical methods such as counting techniques, comparisons with historical data, and reliability modeling. Causal Analysis considers each defect as a unique occurrence and attempts to find the root cause for each defect.

(cont. on p. 2)
Neither of these techniques is universally applied because of the difficulties associated with each of them. Analyzing the number of defects alone is too general to be useful for process improvement, while analyzing individual defects to determine their root cause is time consuming, expensive, and does not always generalize well to future releases. In addition, the analysis and feedback is normally too late to benefit the development project that created the defect. To overcome this problem with causal analysis, many software organizations are instituting end of phase post-mortems. However, this is costly since improvement activities are focused on the entire development process, instead of the "hot spots." Another problem often experienced with these traditional approaches is the inconsistent classification of defects as a result of unclear and overlapping defect classifications. This makes it difficult to do reliable statistical analysis. Finally, these approaches require specially trained staff to perform the analysis.

**Orthogonal Defect Classification**

In order to model the software development process as a controllable and observable system, explicit input-output or cause-effect relationships must be known. Numerous software researchers and industry experts have defined various methods for classifying software defects. Most of the methods developed to date fail to sufficiently quantify the key cause and effect relationships. It is not sufficient to collect a lot of data and hope that some subset of the data will explain the process.

A breakthrough study conducted by Ram Chillarege, et al. (1991), from IBM's Watson Research Center showed that when defects are uniquely categorized by the semantics of the fix, it is possible to relate changes in the shape of the reliability growth curve to defects of a specific type. The IBM study showed that defects of a specific type were the result of a cause in the process that resulted in an effect on the reliability growth curve. This study forms the basis for Orthogonal Defect Classification (ODC).

ODC is a method for classifying and analyzing software defects. It bridges the gap between statistical defect modeling and causal analysis. ODC classifies each defect based upon the semantics of the defect correction and links the defect distribution to the development process and maturity of the product. It provides an in-process measurement paradigm that extracts key properties from defects and enables measurement of cause-effect relationships as opposed to a mere taxonomy of defects for descriptive purposes.

ODC is based upon the principle that different types of defects are normally discovered during different phases of the development life cycle and that too many defects of the wrong type discovered during a particular phase may indicate a problem. In other words, ODC uses the distribution of defect types throughout the software development life cycle to produce a signature for the development process, see Figure 1.

ODC essentially means that software defects are categorized into classes that collectively point to the part of the development process that needs attention. It is used to characterize defects and identify process problems that result in product defects in the same way as x,y,z coordinates are used to identity a point in a Cartesian system of orthogonal axis. "Orthogonal" simply means that the defect classification categories used to characterize a defect don't overlap and are statistically independent of each other.

Chillarege, et al. (1992), defines the following necessary and sufficient conditions for ODC:

**Necessary condition** - There exists a semantic classification of defects such that its distribution, as a function of process activities, changes as the product advances through the process.

**Sufficient condition** - The set of all values of defect attributes must form a spanning set over the development process.

**ODC Defect Attributes**

The challenge for any software defect measurement scheme is to identify a minimal set of defect attributes, in order to keep the classification simple and the overhead added to the development process minimal, while completely mapping all activities of the development process. ODC uses two attributes, Defect Type and Defect Trigger, to provide measurement instruments of the causal relationship of software defects. The Defect Type characterizes the defect based upon the nature of the change to fix the defect. It provides a measurement of the progress of the product through the development process. The Defect Trigger characterizes the defect based upon the catalyst that caused the defect to surface and result in a failure. It provides a measurement of the verification process. A third attribute, Defect Impact, is used to meter the impact of the defect in terms of the effect of the failure on the customer.

**Figure 1 - ODC defect type process signature**

ODC does not imply use of only these three attributes. These attributes should be collected in addition to the traditional defect attributes such as phase found, severity, and missing, incorrect, extra. Any attribute-value set that satisfies the necessary and sufficient conditions can be considered part of ODC. Additional attributes such as the source of the defect, which characterizes the type of code corrected (new, old, reused, vendor, etc.) should be collected along with software size, team size, experience, etc., to characterize the development environment. The key is to create a measurement framework from a minimum set of attributes, that can be sliced various ways to provide visibility into the fundamental issues affecting the software development process, and that is broadly applicable and easily expandable.

**Defect Type**

ODC uses eight categories for defect type: Interface, Function, Build/Package/Merge, Assignment, Documentation, Checking, Algorithm, and Timing/Serialization (see Appendix I). These defect type categories defined by IBM (Chillarege, et al., 1992) apply to any software development project, independent the software product or development process used. The defect type is based upon
the semantics of the defect correction. The categories span all software development life cycle phases, while at the same time each category is associated with a particular development activity. As illustrated by Figure 1, Function defects are typically associated with requirements or high-level design inspections since they constitute the majority of defects found during these phases. While, Timing/Serialization are normally associated with system testing activities.

The relative trend from phase to phase for each defect type category indicates the progress of the product through the development process. For example, see Figure 2, for a typical waterfall development process, a project may be chronologically in the function test phase, but an increase in the percentage of Function defects found relative to coding or design may indicate that the product in actuality should still be in design or coding. This provides an early indication of a process problem only one phase removed from the point in the process that needs attention. With traditional methods the problem would not be identified until integration and system testing or later.

The defect trigger provides a measure of the thoroughness of the verification process.

The defect trigger should not be confused with the symptom of the defect which is the visible effect of a defect that results in a failure. For example, if during an inspection while considering compatibility of the new software with previous versions of the system, a reviewer discovers an assignment error that would result in a particular icon not being properly displayed, the symptom is the icon not being displayed, the trigger is backward compatibility, and the defect type is assignment.

The defect trigger can be used either alone or in combination with other defect attributes. Alone, defect triggers can be used to improve the effectiveness of system testing. For example, the distribution of defect triggers from system test and field/beta testing should be similar. Any difference in the distribution indicates areas of weakness in the system testing. The system testing should be enhanced to provide additional testing using the triggers that resulted in a higher percentage of field/beta testing failures.

![Expected Trend](#)

**Defect Trigger**

ODC uses three sets of defect triggers, Review and Inspection Triggers, Unit and Function Test Triggers, and System and Field Test Triggers, (see Appendix I). For defects found after the software is released to the customer, the trigger is selected from the set of triggers for the testing activity that should have most appropriately detected the defect prior to release. The trigger for a defect is assigned based upon the testing activity that caused the defect to manifest itself as a failure. For example, if during an inspection a defect is discovered as a result of examining the design for compatibility with previous versions of the system the defect trigger would be Design Compatibility.

**Defect Impact**

Defect Impact provides a mechanism to relate the impact of software defects to customer satisfaction. This allows quality improvement efforts to be focused on reducing the defects that most significantly impact customer satisfaction as opposed to blindly reducing the total number of defects. The defect impact is used in addition to defect severity. Severity assesses the magnitude of the failure while impact assesses the capability of the product affected by the defect.

IBM (Kaplan, Clark and Tang, 1994) defines 10 defect impact types, Capability, Usability, Performance, Reliability, Installability, Maintainability, Documentation, Migration, Standards, and Integrity/Security (see Appendix I). These categories were established by IBM based upon customer surveys of what product attributes were most important to their users. Each defect is assigned an impact type based upon the effect that a failure would have on the customer had the defect escaped to the field.

**GSMBSS ODC Feasibility Study**

The GSMBSS Process Improvement Coordination Team (PIC Team) was in search of additional defect prevention and quantitative process management technologies to supplement the Fagan inspection controls already utilized. It was very important that any new technology be piloted and phased into the organization's existing process without a revolutionary upheaval. Thus, the possible synergy of ODC with GSMBSS's Fagan inspection data was especially attractive to the team.

In addition, GSMBSS had been performing statistical correlation between their quality metrics and customer satisfaction scores for over a year. They also had recently introduced a customer Cost Of Ownership Leader (COOL) metric to measure product attributes and services that directly impact the customer's ability to profit from using GSMBSS's products. The customer focus aspect of ODC appeared to provide a vital method to link internal defect prevention activities with the cost of ownership metric and onto external customer satisfaction. The timing was right to integrate ODC into the customer-focused defect prevention strategy if the ODC feasibility study proved successful.

The objective of the feasibility study was to determine:

1. If ODC concepts could utilize GSMBSS's existing Fagan defect classification categories.
2. Confirm that ODC did in fact result in unique process signatures.

Since GSMBSS was already collecting defect data, they did not want to abandon their large database of historical defect data and introduce an entirely new defect classification scheme to their development staff. The plan was to see if it was possible to map GSMBSS's Fagan inspection fault types to the ODC defect types and determine if the resulting process signatures were similar. If this proved plausible, ODC would be considered by GSMBSS for deployment. The final decision rested with the Defect Prevention and PIC Team's review of the technology evaluation report.

**ODC Feasibility Study Results**

The first step of the feasibility study was to map GSMBSS's defect categories to the ODC (cont. on p. 4)
features from completed GSMBSS projects and classified the defects using the ODC fault types. Next we plotted a histogram of the ODC and GSMBSS defects for each development phase and compared the results. Figure 3 shows the ODC defect categories for the three features selected; Figure 4 shows the GSMBSS fault categories for the corresponding features.

Figures 3 shows how the distribution for each ODC defect type changes as the feature passes through the development process and how the distribution is a function of process activities. For example in Figure 1, the percent of assignment type defects found increases from requirements (REQ) to high-level design (HLD) to low-level design (LLD) to code. The percent of interface defects found decreases from REQ to HLD to LLD to code. This is the expected distribution or signature for a typical waterfall development process.

The features used for this study were small. As a result, for an individual feature some phases only had a small number of defects of a particular type, or none at all was discovered. This caused some anomalies in the signatures for the individual features when plotted in terms of percent. For example for feature C, only three defects were found during HLD causing the trend for checking defects to decrease from HLD to LLD. Normally this would indicate a process problem, since it is logical to expect a higher percent of checking errors to be found during LLD and Code than HLD. However in this case, the small number of defects found during HLD causes the trend to be skewed. Therefore, for small projects or features it is important to look at both the percent and absolute number of defects.

Note that even though the trend or signature for each defect type was similar for each feature, the type and number of defects discovered in each phase was different. This is because the number of defects is a function of the complexity and functionality of the product. GSMBSS plans to use historical defect data and product and project attributes in combination with ODC to help calibrate the magnitude of each defect type from phase to phase.

Figure 4 shows the defect data for features A, B, and C using GSMBSS's Fagan Inspection fault types. Two problems with the Fagan Inspection fault types make it difficult to identify defect trends from phase to phase. First, some of the Fagan Inspection fault types only apply during a particular development phase. Second, some of the Fagan Inspection fault types are not orthogonal, i.e., they are not independent and they overlap. This, along with the larger number of categories, makes it difficult to identify a process defect signature in Figure 4 using GSMBSS's Fagan Inspection fault types.

GSMBSS did not want to completely abandon the wealth of historical data they had compiled using their Fagan Inspection fault types. So for the features studied, the authors attempted to map the GSMBSS Fagan Inspection fault types to the single, most appropriate ODC defect type. The resulting ODC defect type distribution is shown in Figure 5.

The expected process signature is more clearly visible in Figure 5 than it was from using the GSMBSS Fagan Inspection fault types, Figure 4. The authors next looked at the actual mapping of the Fagan Inspection fault types to ODC defect types for the three features studied to see if some minor adjustments could be made to the GSMBSS defect types in order to more closely adhere to the ODC principles (Table 2).

Table 2 shows that four Fagan Inspection fault types map to multiple ODC defect types. These four fault types, Cl - Clarification, EH - error handling, LO - logic, MN - Maintainability, account for nearly 50% of the total defects found during development for the features investigated. We recognized that these four Fagan Inspection fault types are actually additional defect attributes that qualify the nature and trigger of the defect type. Clarification...
QUALITY

Figure 4 - “Features A, B, C” GSMBSS defect type vs. phase found

Figure 5 - “Features A, B, C” GSMBSS fault types mapped to ODC defect types

Table 2 - GSMBSS defect type to ODC defect type mapping for features A, B, and C

Handling and Logic are the same as the ODC triggers Recovery/Exception and Operational Semantics, respectively. While Maintainability is an additional defect trigger type not defined by IBM.

Thus, we concluded that with only a few minor modifications, GSMBSS’s Fagan Inspection fault types can be mapped directly into the ODC defect types. This will allow GSMBSS’s PI team to map the Fagan Inspection fault types to the ODC defect types without introducing a completely new defect classification scheme to the GSMBSS software developers. The resulting process signatures from the ODC defect types can be reviewed during development to provide an early indication of the quality of the product. While, the finer level of granularity that the Fagan defect types provide can be maintained to aid in performing root-cause analysis.

ODC FEASIBILITY STUDY FINDINGS

The GSMBSS ODC feasibility study successfully demonstrated that ODC does result in a defect distribution that is a function of the development process and that deviations from the expected “process signature” can be used to provide an early indication of product/process quality problems. The feasibility study also showed that with only a few minor modifications, the GSMBSS’s Fagan Inspection fault types can be easily mapped directly into the ODC defect types.

(Continued on p. 6)
GSMBSS ODC IMPLEMENTATION PLANS

Once GSMBSS's Defect Prevention and PIC Teams were satisfied that the organization's Fagan Inspection fault types could be mapped to the ODC defect types with only a few minor modifications, they were sold on the benefits of deploying ODC. A four-phased implementation plan was developed to deploy ODC, to improve defect analysis and prevention throughout GSMBSS's development life cycle.

The plan was designed to integrate the ODC methodology into the software development life cycle from the beginning and ending phases first and then work inward. Even though GSMBSS was already doing root-cause analysis of post-release defects, the plan was to use ODC to quantitatively focus this analysis on the defects most impacting the customer. At the same time, the plan also included integrating ODC into the front end of the development process to produce in-process defect signatures from inspection data. GSMBSS added ODC in-process defect signatures to their data warehouse plans as a component of feature characterization and modified their end of phase exit criteria to include a review of the in-process defect signatures.

Phase 1 of the implementation plan is currently under way. It involves three parallel efforts: The analysis of post-released defects from 1996 in terms of defect type, trigger, and impact for causal analysis; use of in-process defect signatures by the feature development teams; and use of in-process defect signatures by the Systems Integration and Test teams.

A team of five experienced development engineers representing all GSMBSS product development functional areas was used to categorize all post-release defects from 1996 by Defect Impact. It took them approximately 6 minutes per defect. The results are shown in Figure 6. The team used a subset of IBM's ODC impact types since the impact types cover all system problem areas, not just software. Except for a few minor changes, migration was renamed to expandability and optimisability was added, the ODC impact types defined by IBM mapped directly to GSMBSS's COOL metric.

Another team made up of experience System Integration and Test (SITG) engineers was assembled to determine the defect trigger and type for the set of post-release defects and analyze the results to identify test related corrective actions. The team required less than 3 minutes per defect to assign the defect trigger. The results are shown in Figure 7. The next task for the team is to determine the Defect Type for each defect. An on-line defect reporting form is currently being enhanced to collect the impact and Trigger information in the future.

GSMBSS is currently determining appropriate Fagan Inspection fault type defect signatures commensurate with those published by IBM for ODC. The signatures will be reviewed by the leads of the various feature development tasks to direct in-process corrective actions throughout the design, code, and development test life cycle phases. The PI Team will analyze the results to determine process improvement efforts and to evolve the GSMBSS signatures over time.

The results of the combined defect signatures and Fagan inspection control charts will be reviewed by the SITG prior to the onset of system integration and testing to identify potential problem areas that may require changes to the SITG testing strategy. This may include testing higher risk features sooner or more robust testing of certain features. As escaped defects for each feature are correlated back to the offending feature, the ability to characterize the quality of similar features in the future will evolve.

CONCLUSIONS

This paper has presented an overview of Orthogonal Defect Classification (ODC). In it we showed how ODC can be used to provide process improvement feedback to developers. We also showed how ODC is used to measure the progress of development and to focus improvement activities on the areas that most impact the customer. The principles behind ODC were demonstrated by the results of GSMBSS's ODC feasibility study.

The authors believe that ODC can be easily applied by Motorola software development organizations striving to achieve continuous quality and customer satisfaction improvement. ODC forms an excellent foundation for the development of defect prevention and quantitative process management techniques, similar to those that have been used for years by Motorola's manufacturing operations.

ACKNOWLEDGMENTS

The authors wish to acknowledge the support received from GSMBSS in conducting the feasibility study. Barbara Hirsh, GSMBSS metrics coordinator, was instrumental in supporting this effort. She collected and assisted with the classification and analysis of GSMBSS's inspection data for the feasibility study. Without her help this paper would not have been possible. The authors also acknowledge the strong support by Christine Ioriati, manager of operations, and Vernon Hamlin, manager of operations, who supported the study and fostered follow-on actions to make positive changes based on the results.

Figure 6 - Post-release defect customer impact

Figure 7 - Post-release defect trigger
FUNCTION - The defect was the result of the omission or incorrect implementation of significant capability, end-user interfaces, product interfaces, interface with hardware architecture, or global data structures.

BUILD/PACKAGE AFFECT - The defect was encountered during the system build process, and was the result of the library systems, or with management of change or version control.

ASSIGNMENT - The defect was the result of a value that is incorrectly assigned or not assigned at all. Note that a failure correction involving multiple assignment corrections may be of type Algorithm.

DOCUMENTATION - The problem was the result of an error in the written description contained in user guides, installation manuals, prologues, and code comments. Note this should not be confused with an error or omission in the requirements or design, that might be a Function or Interface defect type.

CHECKING - The defect is the result of the omission of incorrect validation of parameters or data in conditional statements. Note a fix involving the correction of multiple checking statements may be of type Algorithm.

ALGORITHM - The defect is the result of efficiency or correctness problems that affect the task and can be fixed by re/implementing an algorithm or local data structure.

TIMING/SERIALIZATION - The defect is the result of a timing error between systems/subsystems, modules, software/hardware, etc. or is the result of the omission or an incorrect use of serialization for controlling access to a shared resource.

DEFECT TRIGGER - The Defect Trigger is the catalyst that caused a defect to manifest itself as a failure. It provides a measure of the effectiveness of the verification process. Separate triggers are defined for reviews and inspections, unit and function test, and system and field test. Customer reported defects are assigned triggers from all three categories.

Triggers During Review and Inspection Activities

- DESIGN CONFORMANCE - The error was detected while comparing the work product being inspected with the corresponding specification from the prior development phase(s).

- OPERATIONAL SEMANTICS (UNDERSTANDING FLOW) - The error was detected while considering the logic flow needed to implement the function under review.

- CONCURRENCE - The error was detected while considering the synchronization needed between tasks to control a shared resource.

- BACKWARD COMPATIBILITY - The error was detected while considering the compatibility between the function described by the work product under review and that of prior versions of the same product. Note, this requires that the inspector have extensive product experience and familiarity with the function under review.

- LATERAL COMPATIBILITY - The error was detected while considering the compatibility between the function described by the work product under review and other systems/subsystems or functions that it must interface with. Note, this requires that the inspector have broad-based knowledge of the system under development.

- RARE SITUATION - The error was detected while considering some abnormal system behavior that is not specified by the requirements for the function under review. Note, this requires that the inspector have extensive experience and/or product knowledge.

SIDE EFFECTS - The error was detected while considering some system, product, function, or behavior that may occur that is beyond the scope of the work product under review. However, the side effects would be characterized as the result of normal usage or configurations of the system. Note, this requires the inspector to have extensive experience and/or product knowledge.

DOCUMENT CONSISTENCY/COMPLETENESS (INTERNAL DOCUMENT) - The error was detected while reviewing the work product for consistency/completeness and conformance to documentation standards.

- LANGUAGE DEPENDENCIES - The error was detected while reviewing the work product for implementation language specific details.

Triggers During Unit and Function Test Activities

- SIMPLE PATH COVERAGE - The error was detected using White/Gray Box testing to execute simple code paths related to a single function.

- COMBINATIONAL PATH COVERAGE (COMPLEX PATH) - The error was detected using White/Gray Box testing to execute combinations of code paths related to multiple functions. The test case that found the defect was executing combinations of code paths.

- TEST COVERAGE* - The error was detected by using Black Box testing to exercise a single function with either no or a single set of input parameters.

- TEST VARIATION* - The error was detected by using Black Box testing to execute a single function using multiple sets of input parameters, such as illegal values, boundary conditions, and random combinations of parameters.

- TEST SEQUENCING - The error was detected by using Black Box testing to execute multiple functions in a very specific sequence. This trigger applies only if the functions operate correctly when tested independently but fail when executed in a particular sequence.

- TEST INTERACTION* - The error was detected by using Black Box testing to exercise multiple functions in combination. The interaction involves more than a simple sequencing. This trigger applies only if the functions operate correctly when tested independently but fail when executed together.

Triggers During System and Field Test Activities

- WORKLOAD VOLUME/STRESS* - The error was detected while operating the system at or near some resource limit, either upper or lower.

- NORMAL MODE - The error was encountered under normal operating conditions without exercising any particular test suit and with the system operating within resource constraints. This trigger should only be used when system test scenarios cannot be executed because of basic problems that block their execution.

- RECOVERY/EXCEPTION* - The error was detected as a result of executing an exception handler or recovery code. The error would not have been discovered if some earlier event had not caused error handling or recovery processing to be invoked. Note, this trigger is selected only when the error is in the systems ability to recover from a failure, not the failure itself.

- STARTUP/RESTART - The error was detected while the system/subsystem was being initialized or restarted following an earlier shutdown or complete system or subsystem failure.

- HARDWARE CONFIGURATION - The error was detected as a result of testing a particular hardware configuration.

- SOFTWARE CONFIGURATION - The error was detected as a result of testing a particular software configuration.

- TRIGGERS FOR CUSTOMER REPORTED FAILURES - In the case of field reported defects, select a trigger from any of the trigger categories based on what the customer was attempting to do or that best matches the condition that was the catalyst for the failure. For example, if a customer entered a single command with no parameters, the trigger would be Test Coverage. Normally only those triggers marked with an asterisk should be used for field defects.

- DEFECT IMPACT - The Defect Impact provides a mechanism to relate the impact of software defects to customer satisfaction. For defects found prior to release, the Defect

*Indicates may also be used for field defects.
Impact captures information on the effect of a defect on the end user in the event that a defect had escaped to the field and resulted in a failure. For customer reported defects, the Defect Impact captures information on the actual customer impact.

CAPABILITY - The ability of the product/system to perform its intended functions and satisfy the customer's functional requirements.

USEABILITY - The ease with which the product/system can be easily understood and utilized by the customer to perform its intended function.

PERFORMANCE - The speed and responsiveness of the product/system as perceived by the customer.

INSTALLABILITY - The ease with which the product/system can be easily prepared and placed into service.

RELIABILITY - The ability of the product/system to consistently perform its intended functions without unplanned interruption.

MAINTAINABILITY - The ease with which a failure can be diagnosed and the product/system can be upgraded to apply corrective fixes without impacting the customer's data and operations.

DOCUMENTATION - The degree to which the product/system documentation and user's manuals are correct and aid in the customer's understanding and use of the product/system.

MIGRATION - The ease and degree to which the product/system can be upgraded to the newer release without impacting the customer's data and/or operations.

STANDARDS - The degree to which the product/system conforms to established pertinent standards.

INTEGRITY/SECURITY - The degree to which the product/system is protected from inadvertent or malicious destruction, modification, or disclosure.

This paper was chosen as best paper at the 7th International Conference on Software Quality.
ISO/IEC 15504 is an emerging software standard on software process assessment. ISO 15504 incorporates the intent of the ISO 9000 series to provide confidence in a supplier's quality management system. However, it goes far beyond that by providing the acquirer with a framework for assessing software suppliers' capabilities in any of 29 process areas (version 2 of ISO 15504). It also provides guidance on how to carry out process improvement focused self-assessments.

**Relationship between ISO 9000 series and ISO/IEC 15504**

ISO 9001 is a general quality system standard. It has been designed to be used in a third-party audit mode where the organization's quality system is audited against the requirements set out in the standard. If the organization meets all the ISO 9001 requirements (pass-fail mode), it is given a 9001 certificate.

ISO/IEC 15504 incorporates the intent of the ISO 9000 series to provide confidence in a supplier's quality management system. However, it goes far beyond that by providing the acquirer with a framework for assessing software suppliers' capabilities in any of 29 process areas (version 2 of ISO 15504). It also provides guidance on how to carry out process improvement focused self-assessments.

**COURSE REVIEW**

The course was a combination of lectures/discussions (45%) and assessment case study exercises (55%) where the participants worked in groups and presented their findings to the class for review.

There was a rich knowledge of software process engineering among the attendees, which included the developers of SPICE: David Constant, who has thorough knowledge of the SEI Software CMM; Francine Portenier, who worked on the development of ISO 12207 Software Life Cycle Process standard; TickIT trained auditors and ISO 9000 auditors/lead auditors from Canadian and American registrars. The auditors in particular had to learn to become "assessors." There were many rich discussions around the course's curricula and there were constant comparisons between SPICE version 2 and version 3, SEI Software CMM, ISO 12207, and ISO 9000-3.

**Introduction to the ISO/IEC 15504 Reference Model**

The ISO/IEC 15504 (aka SPICE) framework provides a flexible reference model that is scalable to the size and particular processes of an organization. The reference model allows one to focus on any of five process categories, namely customer-supplier (CUS), engineering (ENG), support (SUP), management (MAN), and organizational (ORG), and to assess the capability of each process on a scale of 0 to 5. In the most current version of ISO/IEC 15504 (approved for publication), the processes are strongly aligned with those of ISO/IEC 12207. The capability levels scale of ISO/IEC 15504 includes Incomplete (Level 0), Performed (Level 1), Managed (Level 2), Established (Level 3), Predictable (Level 4) and Optimizing (Level 5). Figure 1 shows how a possibly complex assessment can be expressed with a few informative process capability charts based on ISO/IEC 15504 version 2.

**Example SPICE Rating**

The illustrations at left are a graphical representation of the results of a SPICE assessment for the engineering process category of a single software project. Table 1 defines some of the abbreviations on Figure 1. Figure 1 shows the results of a hypothetical assessment of all seven processes in the process category 'Engineering' for a single software development project. This figure

(continues on p. 10)
WHAT KIND OF JOURNAL DO YOU WANT?

BY TAZ DAUGHTREY

Last newsletter, I shared some results from the recent survey many of you received concerning our new journal-to-be, Software Quality Professional. Those results included background information on your work environment, as well as your perceptions of the division and the Society. This time, I have other results to share from that survey, focusing on your ideas for the journal itself.

The core of SQP's initial readers should come from within the Software Division. We take quite seriously "the voice of the customer." In this case, it is primarily your voice as members of the Software Division. What are you saying?

CONTENT AND FORMAT

As we have seen, you are very supportive of the Society's Certified Software Quality Engineer (CSQE) designation. The scope for SQP will in fact be the Body of Knowledge established for the CSQE. This Body of Knowledge is divided into eight broad areas. The survey asked respondents to rate the relative importance of these areas.

Nearly two-thirds of division members selected "Metrics, measurement, and analytical methods" as their highest-interest area. (It also ranked highest - but not nearly so high - with ASQ members not in the division and non-ASQ members responding to the survey.) SQP plans to cover the full range of CSQE topics, but will certainly seek to emphasize those areas most sought by its readers. The survey indicates that interest is greatest for metrics, followed by quality management, and then inspection, testing, and V&V.

What types of features would be most useful? All populations of respondents put "how-to" articles at the top of the list. Division members also gave relatively high marks to:

- case studies;
- pointers to other resources (Web sites, conferences, etc.);
- summary of research findings;
- and product/tool reviews.

As editor, I will have to be careful because the lowest-ranked categories were editorials, letters to the editor, and (my personal favorite) annotated bibliographies. I'll be sure to go a bit light on those.

HOW ARE THINGS WORKING?

I am ably assisted by five associate editors. The distinguished editorial board is international in composition and broadly representative of all aspects of our profession. Our panel of reviewers operates on the "double blind" principle, meaning reviewers are not told the identities (or affiliations) of authors and they themselves remain anonymous. All relevant submissions are reviewed by at least one member and one nonmember of the editorial board before being examined by the associate editor responsible for that topical area.

Some submissions are solicited. I have personally contacted some of the "names" in our discipline (see next paragraph), who have a proven record of accomplishment. SQP talent scouts attend conferences and scan other publications for interesting topics or promising writers who could be encouraged to offer something to fellow professionals through the journal. Other contributions are delightful surprises—the result of someone seeing information such as this article and believing they have a valuable story to share.

The survey asked you to name those individuals you most respect and admire, both as "quality professionals" and "software professionals." Unsurprisingly, the names of Deming and Juran dominated the first list. A more varied and eclectic roster made up the second list. I have conscientiously approached each of the individuals at the top of that list, and have been delighted that several have already submitted material to the journal. (Of course it is somewhat of a paradox that some are too busy keeping their "most respected" status through work on books and other projects to make any new commitment such as SQP!) I do think you will be pleased to see the lineup of authors we will be publishing.

There were no clear-cut leaders among the dozens of books and quality-related articles named as respondents' favorites, but I gained some additional helpful insights about what appeals to the potential readers of the journal.

YOUR WILLINGNESS TO SERVE

Nearly half of the division respondents indicated they would encourage a colleague to submit or review an article for the journal. More personally, between 30% and 40% also marked "Yes" when asked if they themselves would consider:

- reviewing articles submitted for publication;
- submitting articles for publication; and
- serving on an advisory panel.

OK, I'll take all of you up on that! E-mail me (SQP_Editor@asqnet.org), and I will send you a volunteer's information form. If you wish, you can ask to participate as a reviewer of manu-
scripts, media (books, courses, etc.), or products. If you're interested in submitting material yourself, please check out our "Guidelines for Authors" on http://www.asq.org/sd/sqwweb.html.

Want to propose writing something or maybe just propose the name of another as a potential author? Please send me a short e-mail on the topic, and we can discuss what to do next.

If you want your voice heard—if you have different opinions or if you want to reinforce the views already expressed—please contact me directly. As with any volunteers, "the more the merrier." I am also counting on your feedback once SQA starts publishing, so we can fine-tune this product to better meet your needs. Do stay in touch.

Taz Daughtrey
Division Liaison Chair
Editor-in-Chief, Software Quality Professional

PROVEN METHODS
BY KIM COBLER

This edition's "Proven Method" focuses on the software quality management area. When you are in the process of building a software quality assurance (SQA) area, or if you are faced with adding some vigor to the existing one you should consider the elements found in what I call PEP. PEP helps you build credibility in the SQA area and strengthens the support of senior management and the organization.

Parallel - Maintain a parallel focus of product and process,

Example - Lead the quality effort by example, and

Perception - Use the data to make reality the perception

Parallel - In order to effectively manage software quality assurance there must be equal focus on product quality and process improvement within the organization. This balance improves satisfaction for both internal and external customers.

It doesn't matter how the organization is defined as long as equal attention is given to both areas. If many functional areas have been defined to cover the tasks in the boxes below then you are challenged with effectively integrating the areas to assure balance in activities. If there is one functional area responsible for all of these tasks you are challenged with prioritizing your work to be well balanced.

Balanced attention is important because each customer has different perspectives and values things differently. By addressing a few problems for many customers instead of many problems for a few customers you will increase customer acceptance. Once you have gained acceptance you have to maintain it through action.

Example - Show how committed you are to the principles you are working so hard to implement. Don't get caught in the trap of consistently reacting to the current quality crisis. Ask yourself questions like:

Do I eagerly solicit my customer's input?
Do I have defined metrics for my area and do I consistently track them?
Do I define the requirements for a project and follow them to completion?
Do all of my projects have project plans and work breakdown structures?
Do I create a thorough agenda and publish minutes for all meetings I conduct?
Do I regularly use the seven quality tools to help me perform my job better?
If you aren't doing these things, start. When you are doing all of these things, you'll be proud of your work and the outcomes. So share it with the rest of the organization and change their perception.

Perception - One of my many mentors always used to tell me "Perception is reality." Those words used to haunt me until I finally realized how I could leverage this thought. START CAMPAIGNING! Decide what you want the perception to be. Then, define the necessary goals and metrics to get there. Track your progress, and broadcast the results! The key is using real data, communicating to a large population, setting the organization's expectation, and acting on the outcomes.

Time and time again I hear organizations saying they don't know what QA is or what they do. There is no excuse for that. Start shouting it from house tops or on the Internet/intranet, or on the hallway bulletin board, or through e-mail. The list of communication mediums goes on and on. Use them all and take responsibility for what QA is and does in your organization.

By advertising, performing, and keeping a balanced focus, SQA will thrive. The parallel focus ensures a large population of supporters. Leading by example builds credibility; continuous visibility will create the right perception and make SQA a viable part of any organization.

THE 8TH INTERNATIONAL CONFERENCE ON SOFTWARE QUALITY
will be held jointly with the Pacific Northwest Software Quality Conference on Oct. 12-15, 1998.

Membership Chair Needed

Rusty Perkins, who has served the Software Division in the role membership chair for the past few years, will be resigning that position to fulfill his new duties as regional director on the ASQ board of directors. Software Division members who might be interested in taking Rusty's place should contact Sue McGrath, 919-677-8000 ext. 7032 or sassam@wns.sos.com.
This year's conference was attended by 455 delegates from Washington, Oregon, British Columbia, California, and from as far away as Spain, England, and Australia. As always, this conference represents exceptional value for your conference budget. The major participants are ADP, Boeing, Hewlett-Packard, Intel, Attachmate, Microsoft. World-class speakers are recruited every year, providing authoritative state-of-the-practice papers on all software quality related topics. PNSQC focuses on practitioners, not theoreticians. Internationally recognized speaker and consultant Tim Lister says, "There is no way to lecture at a PNSQC; the audience is too interested and informed to allow anything but a full forum discussion of the topics."

As an attendee and participant at this and many similar conferences over the last 15 years, I vigorously endorse this annual event. Unfortunately I was unable to cover the entire conference this year, so my report will be limited to two outstanding presentations.

EDUCATING SOFTWARE ENGINEERS FOR QUALITY

Terence P. Rout
Software Quality Institute
Griffith University, Queensland, Australia

Terry Rout is a senior lecturer in the School of Computing and Information Technology at Griffith University. He is chair of the Australian Committee for Software Engineering Standards, and has been a member of the international management board for the SPICE project. He is the manager of the Southern Asia Pacific Technical Centre established to provide a regional focus for this work, and is the editor of two of the central components of the SPICE document suite.

Rout opened his keynote address with a quote from Bruce Bond, "There are only two commodities that will count in the 1990s. One is oil and the other is software. And there are alternatives to oil."

Rout stressed, as have many other keynotes at this conference in the past, that, lamentably, academia is not teaching the right stuff. He reminded us of Watts Humphrey's observation that, "We have found that software engineers have difficulty adopting new methods. They first learned to develop software during their formal education and have since followed the same practice with a few adjustments and refinements. Since they are comfortable with these methods and have not seen compelling evidence that other methods work better, they are reluctant to try anything new."

Readers may recall that Mary Shaw (professor of science and associate dean for professional education for Carnegie Mellon), in her keynote address during the 1995 PNSQC, had a very similar lament. Shaw depicted software engineering as falling somewhere between craft and science and recommended that we improve the coupling between science and commercial practice. Watts Humphrey, at the 1996 PNSQC, echoed a similar message when he reported that academia is not teaching engineers to be good programmers, only how to program, thereby releasing the novice engineer into society much like a graduate medical student of surgery who has never done surgery.

Rout's message is a variant on the same theme but with a more far-reaching impact. Rout's message expands the argument by pointing out how pervasive software supported systems are today and how necessary it is to look beyond the offerings of traditional educational curricula. Rout argues that we need a change in the approach to initial education of engineers to help modify the culture. He credits Barry Boehm with succinctly identifying the "Seven Basic Principles of Software Engineering":

1. Manage using a phased life cycle plan
2. Perform continuous validation
3. Maintain disciplined product control
4. Use modern programming practices
5. Maintain clear accountability for results
6. Use better and fewer people
7. Maintain a commitment to improve the process.

Rout cites the following elements as missing from most IT education programs:

- Peer reviews
- Team dynamics
- Measurement principles
- Change management
- Software architecture
- Reuse

He notes that the international computer societies have recognized the need "to define core curricula for accreditation of courses meeting acceptable professional standards," ACM and IEEE have established a joint steering committee for establishing software engineering as a profession. An initial survey outlined 22 distinct knowledge areas that should be addressed by a professional curriculum for IT.

1. Algorithm complexity
2. Computer peripherals
3. Database administration
4. Database systems
5. Distributed systems
6. Power management
7. Risk management
8. Transaction properties
9. Caches
10. Data management
11. Database performance and capacity planning
12. Data structures
13. Effort estimation
14. Project management and planning
15. Software quality assurance
16. Client server
17. Data models for databases
18. Database system fundamentals
19. Device drivers
20. Kernels
21. Real-time systems
22. Static/dynamic linking

In Australia, a draft Body of Knowledge (BOK) for Information Technology Professionals is in work by the Australia Computer Society, which attempts to cover a broad spectrum of information technology areas of which software engineering is only one limited element. The five major "families" within the BOK are:

1. Engineering
2. Computer systems engineering
3. Computer science
4. Information systems
5. Commerce and business administration

Rout sees divergent views emerging among the professional groups and forums, the most significant of which seems to be the blurring of computer science and software engineering objectives.

Interestingly, at Griffith University the basic degree, bachelor of information technology, requires all students to do a one-year project using a Web-based quality manual structured on the new international life cycle standard, ISO 12207. Projects are for real clients drawn from industry, government, and community groups. Informal evaluations show some projects
performing at (SEI) levels 2 and approaching level 3. Success is reported by the clients, not the professor.

Rout believes the move toward software professionalism cannot be reversed. Universities are not graduating enough students to meet the demand. He commented that there are more jobs advertised on one page of a (major) newspaper than are graduated by universities across the country.

His conclusion is that the availability of entry-level personnel properly equipped with the necessary skills (managerial as well as technical) will accelerate the move toward formalized professionalism with long-term benefits to the industry as a whole.

WORLD-CLASS SOFTWARE QUALITY IN PRACTICE

George Yamamura and Gary Wigle
Boeing Information, Space and Defense Systems

George Yamamura is the software engineering process manager for Boeing's Information, Space and Defense Systems (ISDS) in Seattle, WA. He has over 30 years experience in the space application field. He is currently managing the software process organization supporting Boeing programs and working on deployment of best practices. He has degrees in aeronautics and astronautics and applied mathematics. Gary Wigle is a senior principal engineer, also in ISDS, leading the SEPG. He has over 20 years experience in embedded software applications both in the Air Force and at Boeing. He was the SEPG leader for the Space Transportation Systems project, a project that recently achieved SEI level 5 recognition. He has a bachelor's degree in physics from the U.S. Air Force Academy and a master's in systems management from the Air Force Institute of Technology.

This paper discusses how an organization achieved SEI level 5 maturity recognition and the benefits accruing to both the organization, the project, the Boeing Company, and the customer, the Department of Defense (DoD). The authors won the Pacific Northwest Software Quality Conference Excellence Award for their accomplishments.

The Boeing Space Transportation Systems (STS) organization supports DoD and NASA projects. These projects provide main launch systems and upper stage boosters for multiple satellite payloads. Each project has major embedded software components. The assessment leading to level 5 recognition was conducted using the widely recognized, rigorous, and highly reliable SEI CMM-IPI method. It was conducted over a 10-day period by experts in the development of the CMM, Mark Paulk and Steve Masters of the SEI.

Over 40 charts of data were prepared demonstrating the benefits of high maturity accruing to the project and customer. Among these, the authors shared a small but impressive sample. To summarize,

**Defect detection** - from 89% to nearly 100%. A 4% investment in design review returned, 39% in rework reduction, an ROI of 7.75%.

**Productivity** - 100% improved to 240%.

**Performance** - The Inertial Upper Stage (IUS) project has performance criteria consisting of placement of the payload in three dimensional orbit. Deviation from the “bullseye of the targets” requires expending additional fuel to get to final orbit. A percentage of an available “incentive fee” is earned, depending upon hitting progressively smaller diameter rings of the target. Hitting the “bullseye” warrants a 100% incentive fee. Consistent IUS performance not only hit the bullseye, but hit the inner 20% of the bullseye!

Long before the SEI CMM was created, the STS had strong, motivated process champions managing with facts and data. More than 18 years worth of process related data, documents, and standards have been captured in a process library accessed by the program.

The need for process excellence was recognized for the “right” reasons, not just to obtain a maturity score. The right goals consist of a paradigm shift in practitioner and management thinking characterized by four attributes that change focus over the growth period from levels 2 and 3 to levels 4 and 5. These focus changes are:

- From risk to goal management
- From process definition to process measurement
- From practitioner to process management focus
- From local implementation to institutionalization

Boeing's definition of institutionalization goes beyond the traditional view of a process being documented, trained, practiced, and maintained, to being, owned, believed, practiced with pride, and promoted.

An employee satisfaction survey was conducted before process improvement was institutionalized. A 10-point scale was used to measure satisfaction level, from highly dissatisfied to extremely satisfied. The pre-improvement environment was characterized as ad hoc, somewhat chaotic with lots of firefighting teams. Before improvements, 26% were dissatisfied with a mean near the neutral value (5.7). After the improvements, 96% of the workforce reported satisfaction with a mean of 8.3, indicating a very satisfied workforce.

The IUS program has been coined the “University of IUS” for its role in providing trained personnel for other projects. Brochures, policy, plans, procedures, templates, tools, and training are among the process assets available to other projects. In deploying key processes the authors believe there are three key attributes involved:

- Documented plans and procedures, templates
- Having a pool of domain experts
- Key personnel assigned to the SEPG

All three are necessary for successful deployment on new projects. Most important, the authors stress that these actions must be taken for the right reasons. Those reasons are that the processes support the big four business goals: Cost, Schedule, Quality, and Performance. Personnel see how their work supports the business goals, and how they can improve further. An organization-wide SEPG has been formed with the goal of deploying high maturity practices on new programs. This was shown to be achievable on start-up projects included in the level 5 assessment. Probably the most impressive benefit achieved by the IUS project was the cost underrun permitting funds to be given back to the customer! The information shared with the conference attendees earned the authors a standing ovation from an overflow crowd, including myself, who attended this track.

NEW REGIONAL COUNCILORS

BY JAYESH G. DALAL

The council has appointed two new regional councilors; Mike Epner will be the councilor for Region 14 and Armin Torres for Region 15. They will serve for the current term and will be on the ballot for the next term. Mike is in Allen, TX, and Armin is in the Miami, FL area. Both have been active in the field of software quality and with ASQ. Join me in welcoming Mike and Armin to the division council.

We still have open councilor positions for Regions 8, 10, and 11, and the council is looking for volunteers from the division members in those regions.
CALL FOR SUBMISSIONS—PAPERS AND WORKSHOPS

TECHNICAL PAPERS will be presented October 13-14, 1998, and will be published in the conference proceedings and on CD-ROM. Papers should be practitioner oriented. They may be based on research of interest to practitioners, or on experiences that contribute to the software quality body of knowledge. Abstracts are preferred.

WORKSHOP SESSIONS will be held on Monday, October 12, and Thursday, October 15, 1998. Workshops are either half-day or full-day sessions that provide practical knowledge to participants. Workshops that promote the active participation of learners through problem solving, case studies, or other interactive learning methods will be favored.

SUBMISSION REQUIREMENTS: See attached submission cover form, or electronic submission forms are available at www.pnsqc.org or as an auto-response at abstract_form@pnsqc.org.

- **Paper Abstracts**: two- to four-page summary and outline. It should contain enough detail so that the reviewer can estimate the quality of the finished paper and determine if it is fit to our audience.

- **Proposal for a Panel**: One to two pages describing the topic of the panel discussion.

- **Workshops**: Two to five-page summary describing the topics to be presented in the workshop and the presentation methods used (e.g., lectures, case studies, group exercises).

SUGGESTED CONFERENCE TOPICS:
- Software Quality Management
- Software Processes
- Software Project Management
- Software Metrics, Measurement and Analytical Methods
- Software Inspection, Testing, Verification, and Validation
- Software Audits
- Software Configuration Management
- Other Software Quality Related Topics, including Standards, Quality Philosophies and Principles, Organizational and Interpersonal Techniques, Problem-Solving Tools and Processes, and Professional Conduct and Ethics

FOR ADDITIONAL INFORMATION, CONTACT:
(PNSQC) program@pnsqc.org 503-223-8633 (voice)
(ASQ) hunt@eccic.com 407-859-7410 x2306 (voice)

CRITICAL DATES:
March 7, 1998
Abstracts, proposals for workshops and panel sessions due.

April 21, 1998
Notification of acceptance mailed.

June 5, 1998
Complete papers due.

August 14, 1998
Final approved papers due.

PNSQC AND 8ICSQ SUBMISSION COVER FORM
(also available at www.pnsqc.org or send e-mail at abstract_submit@pnsqc.org.)

Return a copy of this completed form as the first page of each submission.

Title of submission:

Primary contact author name:

Affiliation:

E-mail address:

Mailing address:

Work phone: Home:

Fax:

Other authors or contact names:

Affiliation:

E-mail address: Phone numbers:

Key words/phrases:

CHECKLIST:
- Submit author biography (25-50 words)
- Number of pages
- Format: ASCII Text?
- Outline included?

Submission type:
- Paper
- Panel
- Workshop

Intended audience:
- Practitioner
- Process
- Management

Subject scope:
- Practical
- Theoretical

Submit ASCII format to: or Submit 14 paper copies to:
abstract_submit@pnsqc.org.
Theresa L. Hunt
2001 West Oakridge Road
Orlando, FL 32809

March 1998/SOFTWARE QUALITY
**DIVISION COUNCIL MEMBERS**

**Sue McGrath**-Chair  
919-677-8000 ext. 7032  
sassam@vt.sas.com

**Jayesh Dalal** - Chair-Elect  
732-970-0859  
jdalal@lucent.com

**Mike Smith** - Past Chair  
415-604-4449  
mwsmith@mail.arc.nasa.gov

**Sharon Miller** - Treasurer  
732-933-9834  
smiller@northstarcg.com

**G. Timothy Surratt** - Secretary  
630-713-5522  
tsuratt@lucent.com

**Rusty Perkins** - Membership  
Examing Committee  
Internet Liaison  
281-218-2861  
rusty.perkins@lmco.com

**Robin Fralick** - Standards  
610-838-8279  
robin@ptdprolog.net

**Doug Hamilton** - Certification  
847-714-3306  
douglas.b.hamilton@ac.com

**Claire L. Lohr** - Education & Training  
703-391-9007  
lohrsys@erols.com

**Barbara Frank** - Publications  
972-519-3297  
bfrank@ccmail.ccsc.com

**Rick Biehl** - Strategic Planning  
407-296-6000  
70573.1544@compuserve.com

**Linda Westfall** - Programs  
972-519-3779  
lwestfall@ccmail.ccsc.com

**Taz Daughtrey** - Liaison, Journal Editor  
804-522-5137  
sqt_editor@asqnet.org

**Tom Griffin** - ICOSQ  
334-244-3304  
tgriffin@monk.aum.edu

**David Zubrow** - Metrics, Measurement and Analytical Methods  
412-268-5243  
dz@sei.cmu.edu

**Kim Cobler** - Methods  
801-363-7770  
cobler@burgoyne.com

**John Pustaver** - Awards  
508-433-4254  
pustaver@swquality.com


**REGIONAL COUNCILORS**

**Region 1** - John Pustaver  
508-433-4254  
pustaver@swquality.com

**Region 2** - Jean Burns  
607-779-7868  
bums@uic.com

**Region 3** - Bill Fokom  
203-385-4339  
bvr3919@dcdr.dla.mil

**Region 4** - Stephen White  
613-592-7400 ext. 2088-w  
swhite@kan.marconi.ca

**Region 5** - Joel Glazer  
410-765-4567  
glazer.joel@postal.essd.northgmm.com

**Region 6** - M.P. Kress  
206-266-0545  
michael.p.kress@boeing.com

**Region 7** - Harry Ohls  
818-798-1160

**Region 8** - Open

**Region 9** - John Lowe  
513-429-6658  
lowejohn@prc.com

**Region 10** - Open

**Region 11** - Open

**Region 12** - Bob Colby  
630-979-6783  
rcolby@lucent.com

**Region 13** - Paul Caraccio  
303-987-5499, ext. 16  
paul@censi-dev.com

**Region 14** - Mike Epner  
972-390-1576  
sqt@compuserve.com

**Region 15** - Armin Torres  
305-380-3768  
amint@msn.com

**Region 25** - Martin Swafford  
214-363-6181, ext. 411  
swafford@autotester.com

---

**Regional Map**

**4** = Canada

**6** = Mexico

**25** = International

**Includes Mexico**
CSQE Exam
October 17, 1998
Application deadline is August 21, 1998
Call ASQ for information

52nd AQC
May 4-6, 1998
Philadelphia, PA

Software Division Council Meeting
AQC
May 3, 1998
9:00 a.m. to 5:00 p.m.

Software Division Session
AQC
May 5, 1998
“Bad Software—Who is Liable?”
3:30 p.m. to 5:30 p.m.

Submit articles for the next issue of Software Quality by May 1, 1998.
Barbara Frank
Phone: 972-519-3297
Fax: 972-519-2618
e-mail: bfrank@ccmail.dssc.com

WANTED
Regional Councilors for Regions 8, 10, and 11
Those interested in these positions should contact:
Jayesh Dalal
Phone: 908-949-6703
e-mail: jdalal@lucent.com

ASQ CERTIFICATION
A Mark Of Excellence
ENHANCE your career and self-esteem
ADVANCE within your organization
REALIZE your salary goals
AFFIRM your commitment to quality


Priority Code CTAP5D8