Definitions and processes related to software quality must be standardized, according to most participants in the recent ACM Software Quality Assurance Workshop. The need for such standards will become even more apparent over the next five to eight years, since the Department of Defense is expected to require their use for software acquisition.

An effort towards standardization of the process of software quality assurance is already under way with the development of proposed IEEE standard P730, which unifies various software QA techniques and practices. The IEEE standard and two other promising techniques—software configuration management and the systematic analytic modeling method—were among topics discussed at the workshop, held November 15-17 in San Diego.

The three-day workshop drew 280 attendees spanning the globe from New Zealand to Norway. Participants came from a mix of large and small vendors with defense vendors highly represented. Thirty-one papers were presented describing various aspects of quality assurance from the design of software quality at the requirements level to the measurement of quality at the code level.

IEEE standard unifies QA techniques, practices

The need for standards in software quality assurance plans was outlined by F. J. Buckley, chairman of the Computer Society’s Software Engineering Standards Subcommittee, which is responsible for proposed IEEE Standard P730. 1 Buckley addressed one obvious motivation for the standard: software failure can—and does—cause large-scale social and financial loss. He cited a recent Florida case in which a man’s vehicle was incorrectly identified by law enforcement computers as being stolen. During attempted apprehension, he was killed by police gunfire. Clearly, this and other cases like it in our age of product liability produce a financial—and moral—obligation to quality assure critical software.

P730 provides a schema to bring together various software QA techniques and practices into a unified whole. The standard identifies documentation, reviews and audits, and configuration management as areas that must be planned for and executed to produce quality software. Proposed minimum software documentation would include a software requirements specification (SRS), software design description (SDD), and software verification plan (SVP). The documents would then be reviewed and audited as follows: the software requirements review of the SRS, the preliminary design review of the SDD, and the critical design review of the SVP. These documents and their corresponding reviews would form the basis for ensuring that requirements are correctly translated into code.

Other elements of P730 provide for coding practices and conventions, configuration management, problem reporting, test tools and verification, and code and media controls. These elements could be considered the production methodology used to create and test the software. For coding practices and conventions, the standard identifies various methods for compliance (e.g., programming languages, commentary standards, analytical verification methods, etc.); for the other elements of the standard, no techniques or methodologies have been suggested due to the lack of consensus on approaches. However, Buckley pointed out, volunteers are being sought to help revise and extend the standard.

SCM introduces discipline to software management

Software configuration management is a means of “introducing discipline to the software management process,” according to Stan Siegel, who presented a paper on SCM. 2 Typical software development problems, such as schedule delays and cost overruns, yield to the discipline of SCM, which provides visibility and substance to the software development process so that management can track and respond to functional changes to the software.

The roots of SCM lie in hardware configuration management, a technique that has been used successfully for some time. The correspondence in intent can be stated as, “SCM, like CM, is defined as the discipline of identifying the configuration of a system
Auditing. Each baseline and its subsequent updates are SCM components subject to the audit function, which verifies and validates all five baselines in a software configuration. For example, the functional baseline audit examines the linkages between system requirements and software requirements; and the product baseline audit assures that executable forms of an SCI perform according to requirements.

Samm clarifies customer, supplier communications

Whatever management methods are used to ensure quality software, customer software requirements must be accurately transmitted to the coded and installed product. Toward this end, a methodology known as systematic activity modeling method has been developed. SAMM seeks to clarify communications during the software development life cycle (defined here as problem definition, requirements, specification, design, code, test, installation, and maintenance) particularly at the software requirements, software design level.

Motivation for clarifying communications at the requirement-design interface comes from observations such as Horst Rittel’s that “a statement of a problem is a statement of the solution.” Thus, when a customer presents a problem, or requirement, he already has at least a partial solution. Unfortunately, the customer’s conception of the problem/solution often is not accurately transmitted to the designers.

Acknowledgments

I would like to thank General Chairman A. C. (Ton) Shetler, Robert Patrick, Mort Bernstein of the program committee, and Stan Siegel for their informative counsel.

References


3. op. cit., pg. 11.
