The Consortium for IT Software Quality is a sponsored special interest group managed by the Object Management Group (OMG). CISQ is chartered to create international standards for measuring the size and structural quality of software from analysis of the source code. CISQ’s mission is to dramatically increase the use of software product measures in software engineering and management. The Executive Director of CISQ, Dr. Bill Curtis, is best known for leading the development of the Capability Maturity Model for Software (CMM) while at the Software Engineering Institute.
CISQ was formed in 2010 when both the Software Engineering Institute at Carnegie Mellon University and OMG were approached by system integrators and asked to develop standards for measuring software attributes such as reliability and security. These attributes were appearing in development and outsourcing contracts as the equivalent of service level agreements, and every customer had a different definition of how they were to be measured. SEI and OMG co-founded CISQ and asked Dr. Bill Curtis who had led development of the original Capability Maturity Model or CMM to head the consortium. Twenty-four companies including both customers and vendors joined to create the first round of measurement standards. CISQ has evolved to OMG’s traditional model of a special interest group with several companies sponsoring CISQ’s activities. Current sponsors are Accenture, Cognizant, Booz Allen, Huawei, Synopsys, and CAST. Under this structure, membership in CISQ is free for individuals, and I will show you how to become involved at the end of the talk.
At its launch, CISQ held executive workshops in Washington DC, Frankfurt Germany, and Bangalore India. During these workshops we asked IT executives what measures they wanted CISQ to develop. They selected five measures. First, they requested a standard for automating Function Points that mirrors the IFPUG counting guidelines. However, automation requires that the ambiguity and inconsistency characterizing manual counting be eliminated. Second, they selected four measures of structural quality—Reliability, Performance Efficiency, Security, and Maintainability. Experts from the 24 original member companies worked for two years to create automatable specifications, which were then submitted to OMG’s standards approval process. The Automated Function Point measure was approved as an OMG standard in 2013, and the 4 quality characteristic measures were approved as OMG standards last year. CISQ holds periodic workshops to introduce these measures to the IT community. CISQ has also begun entering these measurement standards to OMG’s expedited submission process to become ISO standards.
CISQ’s first published standard was a specification for Automated Function Points. The specification team was led by David Herron, an acknowledged expert in functional measurement, co-founder of the David Consulting Group, and a leader in the International Function Points User Group (IFPUG) community. David led an international team in creating a specification that mirroring the IFPUG counting guidelines while removing ambiguity in the counting rules to enable automation. This specification was approved as an OMG standard in 2013 and is now available in vendor supported technologies. It has been adopted by numerous companies and the French government requires it for reporting size on government contracts.
CISQ’s four structural quality characteristic measures are based on quantifying violations of good architectural and coding practice within a software system that can be detected through static analysis. Violations were included in each measure only if they were considered severe enough that they should be eliminated. Shown here are the number of violations included in each measure and three example violations. For instance, the Reliability measure consists of 29 violations that can cause outages or erratic behavior. Examples include empty exception blocks, unreleased resources, and circular dependencies. Most measures of Reliability assess system availability or downtime which are behavioral measures. The CISQ measures assess actual flaws in the software that can cause operational problems. Thus, the CISQ measures provide pre-release indicators operational or cost of ownership risks.
Let’s look more deeply at the CISQ Security measure. The Security measure effort was led by Bob Martin who oversees the Common Weakness Enumeration Repository maintained by MITRE Corporation. This repository contains over 800 known weaknesses that hackers exploit to gain unauthorized entry into systems. Periodically the assurance community determines the Top 25 weaknesses, and they become the basis for the CWE/Sans Institute Top 25 Most Dangerous Security Errors and the OWASP Top 10 Vulnerabilities. Of these top 25 weaknesses, 22 can be detected through static analysis of the source code. These 22, listed here in numerical order by their CWE identifiers, became the basis for the CISQ measure. Among the most exploited weaknesses are the perennial favorites: SQL injection, cross-site scripting, and buffer overflows. We have known about SQL injection since the late 1990s. How can this weakness continue to be a common entry for hackers? Too many IT organizations have failed to cleanse their systems of obvious weaknesses.
Measuring the structural quality of a modern business application is tricky. Most modern applications are developed as a stack of technologies—multiple layers written in multiple technologies. The user interface may be written in ASP or JSP, the business logic in Java or .NET, data queries in SQL, and if you are in a bank, you will probably be interacting with a backend written in COBOL. Modern systems need to be analyzed at three levels. At the Code Unit level, components are analyzed by developers to ensure good code hygiene. At the Technology Level, many components written in the same language are integrated requiring analysis of a thicket of components across an application layer. At the System Level different technology layers are integrated into the application system. The complexity of these multilayer, multi-language systems exceeds the capability of any single individual or team to understand the whole of it. Consequently developers make assumptions, some incorrect, about how components at different levels will interact. Operational fiascos frequently result from unintended interactions across different layers.
The quality of a modern business application cannot be determined unless analysis includes the system level. For instance, you cannot assess the Security vulnerability of a multi-layer system by merely assessing whether there is an authentication capability. Analysis at the system level can detect potential entries to the system that elude the authentication layer. Analysis at the system level is challenging, especially when it involves multiple languages and platforms. Detection of some security weaknesses requires an ensemble of analysis methods. System level analysis technology will be evolving over the next several years to provide greater capability for detecting weaknesses.
To the extent possible, CISQ aligns its measures with other international standards. For instance, the four CISQ quality measures are aligned with definitions provided in ISO/IEC 25010, the new standard defining software product quality. The four CISQ quality measures map directly to four of the quality characteristics in 25010. Each quality characteristic in 25010 is divided into subcharacteristics that indicate its scope of coverage. These subcharacteristics guided the selection of violations for each CISQ measure. Thus, CISQ ensured its measures covered the related domain described in 25010. The standard that defines the actual measures of each quality characteristic is ISO/IEC 25023. Dr. Curtis is a co-editor of this standard, but he was unable to convince them to define measures at the source code level. Consequently, 25023 primarily defines measures at the behavioral level, such as downtime rather than empty exception blocks. Since CISQ provides measurement at the source code level, it supplements ISO/IEC 25023.
The original request that launched CISQ was for international standards for measures of structural quality that could be used in software contracts. The CISQ size and quality characteristic measures can be written into contracts and Service Level Agreements to provide a well-defined, common standard for setting targets and contractual agreements. CISQ measures can then be used to track actual performance against agreed targets, as well as being aggregated into other management reports for tracking vendor performance. There is a white paper inside the member’s area of the CISQ website that describes how to use CISQ measures in contracts.
The continuing stream of multi-million dollar failures is causing an increased demand for certifying software. Although CISQ will not provide a certification service, it will provide an assessment process to endorse that a technology can detect the weaknesses that comprise the CISQ Quality Characteristic measures. Vendors using CISQ-endorsed technologies can certify the level of quality in a software application using the CISQ measures. These certifications do not guarantee incident-free performance, rather they attest the level of quality in the software, reported in sigma levels, with an option to report defect densities. The vendors must use people who have been trained in the technology and the CISQ weaknesses to perform analyses and deliver certification results. CISQ has announced a program for evaluating the compliance of vendor technologies.
I promised at the beginning that I would show you how to become more involved in CISQ. This is the CISQ homepage. To become a member, just click the ‘Member’ button below the nice lady. You will be asked a few questions and we will send you a password. Membership is free thanks to our sponsors. Inside you will find the definitions of the various measures, white papers describing how to use them or incorporate them into contracts, presentations given at conferences, and other resources. We invite you to contribute to the member blog. Shortly we will be surveying the membership to gather data on how long it takes to fix each of the violations in the CISQ measures so that we can create a measure of technical debt that predicts future costs of corrective maintenance. Next year we will begin extending the CISQ quality characteristic measures to embedded software. I invite you to join CISQ and help us develop a community focused on improving the structural quality of software-intensive systems.