EDITOR RICCARDO MARIANI
NVIDIA; rmariani@nvidia.com

Measuring the Structural Quality of Software Systems

Bill Curtis, Consortium for Information and Software Quality
Robert A. Martin, MITRE Corporation
Philippe-Emmanuel Douziech, CAST

The International Organization for Standardization recently published ISO/IEC 5055:2021 for measuring the reliability, security, performance efficiency, and maintainability of software systems by detecting and counting severe violations of good architectural and coding practice.

In *Code Quality*,[1] Diomidis Spinellis observed that "…a failure to satisfy a nonfunctional requirement can be critical, even catastrophic…nonfunctional requirements are sometimes difficult to verify. We cannot write a test case to verify a system's reliability." Consequently, he concludes, "The ability to associate code to nonfunctional properties can be a powerful weapon in a software engineer's arsenal." As Spinellis implies, attributes such as reliability and security are primarily affected by the structural rather than functional characteristics of a software system's architecture and code.

Standards regarding the structural quality of software systems such as CERT C[2] and the Motor Industry Software Reliability Association system[3] have typically focused on embedded system languages, such as C and C++. Their use has generally been in industries such as automotive and avionics that embed microchips in their products. ISO/IEC 25023[4] defines software quality measures, but they primarily quantify the operational behaviors and outcomes that result from the quality of the software, rather than measuring structural attributes that cause the behaviors. Consequently, industry has needed an international standard defining measures derived from source code analysis covering both embedded and business systems.

Digital Object Identifier 10.1109/MC.2022.3145265
Date of current version: 11 March 2022

ISO/IEC 5055:2021

To meet this need, the International Organization for Standardization (ISO) published ISO/IEC 5055:2021

Automated Source Code Quality Measures[5] in March 2021. This standard was developed initially by the Consortium for Information and Software Quality (CISQ) with contributions by experts from 31 companies in North America, Europe, and Asia. CISQ was cofounded by the Software Engineering Institute at Carnegie Mellon University and the Object Management Group (OMG) to create standards for automating software measurements. After the initial specification was approved by the OMG, it was submitted to ISO and approved as a publicly available standard.

ISO/IEC 5055 defines four structural quality measures derived from static analysis of a software system's source code and architectural structure. These four measures assess the extent to which a software system is free from severe weaknesses that could affect its reliability, security, performance efficiency, and maintainability. These four quality attributes were prioritized by software executives invited to meetings held in Washington, D.C.; Frankfurt, Germany; and Bangalore, India.

The measures can be normalized by size to indicate defect density, by failed checks against opportunities to assess rule compliance or sigma level, or other comparable metrics.

ISO/IEC 25010:2011[6] enumerates eight quality characteristics, four of which map to the four measures in ISO/IEC 5055. ISO/IEC 5055 adhered to the definitions of the four quality characteristics in ISO/IEC 25010 and used their subcharacteristics to ensure domain coverage by the weaknesses composing each measure. Figure 1 presents the ISO/IEC 25010 quality model with the quality characteristics covered by ISO/IEC 5055 shaded in blue.

CALCULATING MEASURES FROM WEAKNESSES

The international team convened by CISQ sorted through a wide range of software weaknesses and selected those most severe for inclusion in ISO/IEC 5055 measures. For purposes of selecting weaknesses, the assessment "severe" was based on believing a weakness had to be removed from the software to avoid damaging operations or excessive cost of ownership. Since many of the most severe weaknesses involve erroneous interactions among components spread across the stack of technologies composing a system, weaknesses were defined using a language-independent representation.

The four measures are calculated by counting the number of weaknesses detected for each of the four quality characteristics. These counts can then be

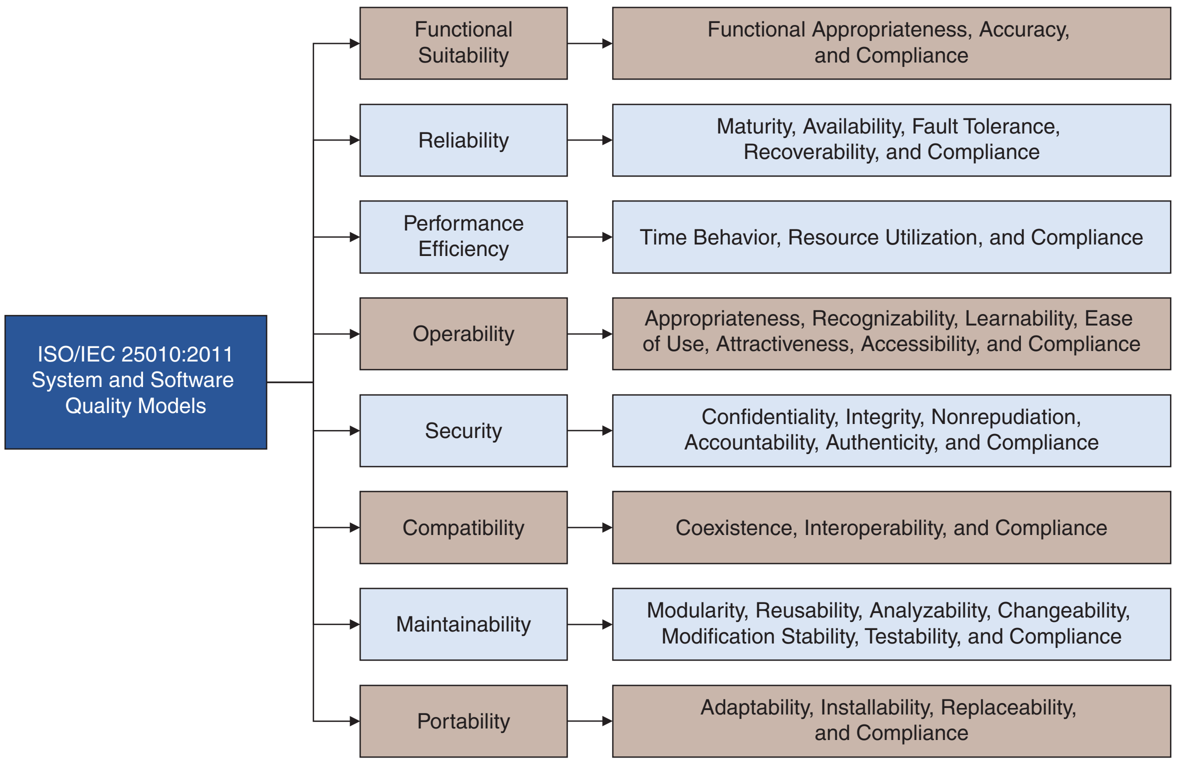
ISO/IEC 25010:2011 System and Software Quality Models
Functional Suitability
Functional Appropriateness, Accuracy, and Compliance
Reliability
Maturity, Availability, Fault Tolerance, Recoverability, and Compliance
Performance Efficiency
Time Behavior, Resource Utilization, and Compliance
Operability
Appropriateness, Recognizability, Learnability, Ease of Use, Attractiveness, Accessibility, and Compliance
Security
Confidentiality, Integrity, Nonrepudiation, Accountability, Authenticity, and Compliance
Compatibility
Coexistence, Interoperability, and Compliance
Maintainability
Modularity, Reusability, Analyzability, Changeability, Modification Stability, Testability, and Compliance
Portability
Adaptability, Installability, Replaceability, and Compliance

FIGURE 1. Coverage of ISO/IEC 5055 measures in the ISO/IEC 25010 software quality model.

normalized for use in benchmarking or trend analysis. The measures can be normalized by size to indicate defect density, by failed checks against opportunities to assess rule compliance or sigma level, or other comparable metrics.

The four measures are constructed from a list of 138 unique weaknesses, examples of which are presented in Figure 2. All 138 weaknesses are contained in the Common Weakness Enumeration (CWE) repository[7] maintained by MITRE Corp (cwe.mitre.org). Weaknesses are divided between 92 primary weaknesses and 46 contributing weaknesses. Contributing weaknesses encompass various structural patterns through which 13 of the primary weaknesses can be instantiated in source code.

The ISO/IEC 5055 weaknesses include serious flaws at both the architectural and component levels to provide a broad evaluation of the factors determining a system's integrity. For example, CWE-424: Improper Protection of Alternate Path is an architectural weakness that violates security and data protection controls by allowing a path from the user interface directly to the database without passing through user authentication routines. CWE-404: Improper Resource Shutdown or Release is a reliability and performance efficiency weakness that has frozen customer-facing systems during critical business hours. The other weaknesses comprising ISO/IEC 5055 have similar undesirable impacts on business operations and cost of ownership.

Fifty weaknesses overlap two measures and are included in the calculation of each. The impacts of six weaknesses are so extensive that they impact three measures. The most extensive overlap of weaknesses occurs between the categories of reliability and security with 38 weaknesses included in the calculation of both. This overlap occurs because some weaknesses causing reliability problems can also create opportunities for unauthorized access.

REPRESENTING ISO/IEC 5055 WEAKNESSES

All 138 weaknesses are represented in metalanguages to guide static analysis vendors in automating their detection. The formal description provides an overview of the abstract weakness pattern and computational entities playing a role in the pattern. It also lists the uniform resource locator where the weakness can be found in the CWE repository and the detection patterns needed for guiding implementation in static analyzers.

The standard then enumerates 135 detection patterns that provide guidance for detecting the weaknesses in source code. The detection patterns are represented in the Micro Knowledge Discovery Metamodel (KDM, ISO/IEC 19506[8]), an intermediate representation of the metadata and computational elements emerging from parsing source code. The detection patterns are essentially a pseudocode representation of the structural elements of a weakness. Each unique instantiation of a weakness is represented in a detection pattern, and some weaknesses are associated with more than one detection pattern.

For example, one of two detection patterns for CWE-672: Operation on a Resource after Expiration or Release would identify occurrences in the code where a path from the resource release statement leads to the resource access statement excluding pointers. The Micro KDM representation of this detection pattern is as follows:

```
PlatformModel
  …
  DataManager|FileResource id=“pr1”
  …
  PlatformResource id=“pa1”
    kind=“open”
    implementation=“ae4”
    ManagesResource “pr1”
  PlatformResource id=“pa2”
    kind=“close”
    implementation=“ae1”
    ManagesResource “pr1”
…
CodeModel
  …
  ActionElement id=“ae1”
    kind=“PlatformAction”
    Flows “ae3”
  ActionElement id=“ae3”
    Flows “ae4”
  ActionElement id=“ae4”
    kind=“PlatformAction”
```

USING ISO/IEC 5055 MEASURES

ISO/IEC 5055 measures can be used to set measurable targets for sustaining

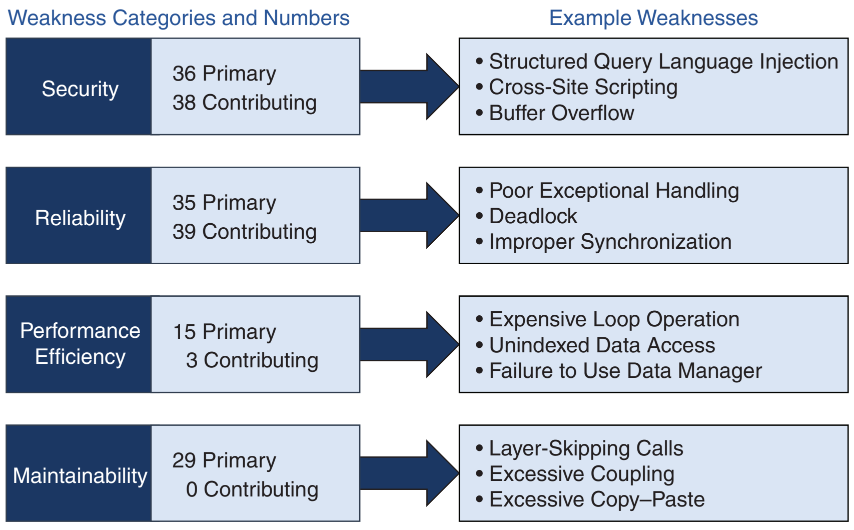

FIGURE 2. Example weaknesses for each of the four ISO/IEC 5055 measures.

the reliability, security, performance efficiency, and maintainability of software systems. These targets can be written into requests for proposal, statements of work, and contracts as acceptance criteria for software products delivered by system integrators, software vendors, and other third-party suppliers. They can also be used with internal software teams to establish release criteria or improvement targets. Some weaknesses contained in ISO/IEC 5055, such as the most dangerous security and reliability weaknesses, can be marked as "unacceptable," and software should not be put into operation until they have been removed.

The severity of a weakness depends on the context of its position in a software system. In some contexts, a severe weakness can become less onerous, while a less severe weakness may become more dangerous. The weaknesses in ISO/IEC 5055 measures were selected because they expose systems to substantial operational and cost of ownership risk in most contexts. However, the severity of individual weaknesses can be assessed using the Common Weakness Scoring System[9] to help prioritize corrective actions.

ISO standards undergo a systematic review every five years. This provides ISO/IEC 5055 an opportunity to update the list of weaknesses in each measure. New classes of severe weaknesses can be added. Empirical evidence from operational and cost of ownership research can cause others to be removed. Thus, ISO/IEC 5055 will be periodically updated as computing technology and languages evolve. ◼

ACKNOWLEDGMENTS

We thank the approximately 80 software engineering professionals and their CISQ member sponsors for participating in selecting the weaknesses and developing the measurement specification that ultimately became ISO/IEC 5055. We also thank the OMG Architecture Board for reviewing the specification and improving its presentation. Finally, we acknowledge the help of David Filip of ADAPT Center in Ireland for guiding us through the ISO approval and publication process.

REFERENCES

1. D. Spinellis, *Code Quality: The Open Source Perspective*. Upper Saddle River, NJ, USA: Addison-Wesley. 2006, pp. xxvi–xxvii.
2. Software Engineering Institute, "SEI CERT C coding standard: Rules for developing safe, reliable, and secure systems (2016 Edition)," Carnegie Mellon Univ., Pittsburgh, PA, USA, 2016. [Online]. Available: https://resources.sei.cmu.edu/library/asset-view.cfm?assetID=454220
3. "MISRA C:2012 Third Edition, First Revision," The MISRA Consortium Limited, Norwich, U.K., 2012. [Online]. Available: https://www.misra.org.uk/product/misra-c2012-third-edition-first-revision
4. *Systems and Software Engineering — Systems and Software Quality Requirements and Evaluation (SQuaRE) — Measurement of System and Software Product Quality*, ISO/IEC 25023:2016, International Organization for Standardization, Geneva, Switzerland, 2016. [Online]. Available: https://www.iso.org/standard/35747.html#:~:text=ISO%2FIEC%2025023%3A2016
5. *Information Technology — Software Measurement — Software Quality Measurement — Automated Source Code Quality Measures*, ISO/IEC 5055:2021, International Organization for Standardization, Geneva, Switzerland, 2021. [Online]. Available: https://standards.iso.org/ittf/PubliclyAvailableStandards/
6. *Systems and Software Engineering — Systems and Software Quality Requirements and Evaluation (SQuaRE)* — System and Software Quality Models, ISO/IEC 25010:2011, International Organization for Standardization, Geneva, Switzerland, 2011. [Online]. Available: https://www.iso.org/standard/35733.html
7. R. A. Martin, "Managing vulnerabilities in networked systems," *IEEE Softw.*, vol. 34, no. 11, pp. 32–38, Nov. 2001. doi: 10.1109/2.963441.
8. *Information Technology — Object Management Group Architecture-Driven Modernization (ADM) — Knowledge Discovery Metamodel (KDM)*, ISO/IEC 19506:2012, International Organization for Standardization, Geneva, Switzerland, 2012. [Online]. Available: https://www.iso.org/standard/32625.html
9. "Common Weakness Scoring System (CWSS)," MITRE Corp., McLean, VA, USA, 2014. [Online]. Available: https://cwe.mitre.org/cwss/cwss_v1.0.1.html

BILL CURTIS is executive director of the Consortium for Information and Software Quality, a software measurement standards consortium cofounded by the Object Management Group and Software Engineering Institute. He is a Fellow of IEEE. Contact him at curtis@acm.org.

ROBERT A. MARTIN is a senior principal software and supply chain assurance engineer at MITRE Corporation. Contact him at ramartin@mitre.org.

PHILIPPE-EMMANUEL DOUZIECH is a principal research scientist at CAST Research Labs focused on machine learning applied to software structural analysis Contact him at philippe-emmanuel.douziech.1992@mines-paris.org.