Automating Software Quality Measurement with Standards

Dr. Bill Curtis
Founding Executive Director, CISQ
CISQ Webinar — May 15, 2019
The Era of Nine-Digit Defects

Nine Digit Defects now affect

Board of Directors
CEO, COO, CFO
Business VPs
Corporate Auditors
CIO

accountable for

Governance
Risk management
Business Continuity
Brand protection
Customer experience

Evaluate Application Risk with CISQ Measures
Why Do Software-Intensive System Projects Fail?

### Failure causes
- Unrealistic expectations
- Incomplete requirements
- Weak contracts & control
- Unachievable schedules, budgets
- Changing requirements
- Unsound architecture & coding
- Staff inexperience & turnover
- Truncated testing
- Botched deployments

### Failure avoidance practices
- Feasibility analysis
- Mission & business analysis
- Acquisition management
- Project & risk management
- Baseline management
- Technical risk measurement
- Contractual personnel clauses
- Quality management
- Acceptance management

Software measurement
Modern Apps are a Technology Stack

1. **Unit Level**
   - Code style & layout
   - Expression complexity
   - Code documentation
   - Class or program design
   - Basic coding standards
   - Developer level

2. **Technology Level**
   - Single language/technology layer
   - Intra-technology architecture
   - Intra-layer dependencies
   - Inter-program invocation
   - Security vulnerabilities
   - Development team level

3. **System Level**
   - Integration quality
   - Architectural compliance
   - Risk propagation
   - Application security
   - Resiliency checks
   - Transaction integrity
   - Function point
   - Effort estimation
   - Data access control
   - SDK versioning
   - Calibration across technologies
   - IT organization level
What Is the Consortium for IT Software Quality?

Co-founders

Dr. Paul Nielsen, CEO
Dr. Richard Soley, CEO

Develop specifications for automatable measures of software systems

Gain approval as OMG standards

Fast-track to ISO
CISQ/OMG Standards Process

- Automated Function Points
- Reliability
- Performance Efficiency
- Security
- Maintainability

OMG Approved Standards

OMG

ISO Fasttrack

Deployment Workshops
Automated Size Measurement

- Mirrors IFPUG counting guidelines, but automatable
- Specification developed by international team led by David Herron of David Consulting Group
- Submitted thru OMG’s fasttrack as ISO 19515
An international team of experts selected the weaknesses to include in CISQ measures based on the severity of their impact on operational problems or cost of ownership.

Only weaknesses considered severe enough they must be remediated were included in the CISQ measures.

CISQ Structural Quality measures have been extended to embedded systems software.
CISQ Measures Updated for Embedded Systems

- With all the functionality being embedded on chips, the line between embedded and IT software is blurring
- All CISQ weaknesses are now identified with CWE numbers (ITU-T X.1524; UN standards body)
- Some CISQ weaknesses presented in parent-child relationships
- Attempting to get CISQ quality measures referenced in revision of ISO/IEC 25023

<table>
<thead>
<tr>
<th>CWE #</th>
<th>Descriptor</th>
<th>Weakness description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CWE-22</td>
<td>Improper Limitation of a Pathname to a Restricted Directory (&quot;Path Traversal&quot;)</td>
<td>The software uses external input to construct a pathname that is intended to identify a file or directory that is located underneath a restricted parent directory, but the software does not properly neutralize special elements within the pathname that can cause the pathname to resolve to a location that is outside of the restricted directory.</td>
</tr>
<tr>
<td>CWE-23</td>
<td>Relative Path Traversal</td>
<td>The software uses external input to construct a pathname that should be within a restricted directory, but it does not properly neutralize sequences such as &quot;..&quot; that can resolve to a location that is outside of that directory.</td>
</tr>
<tr>
<td>CWE-36</td>
<td>Absolute Path Traversal</td>
<td>The software uses external input to construct a pathname that should be within a restricted directory, but it does not properly neutralize absolute path sequences such as &quot;/abs/path&quot; that can resolve to a location that is outside of that directory.</td>
</tr>
<tr>
<td>CWE-77</td>
<td>Improper Neutralization of Special Elements used in a Command (&quot;Command Injection&quot;)</td>
<td>The software constructs all or part of a command using externally-influenced input from an upstream component, but it does not neutralize or incorrectly neutralizes special elements that could modify the intended command when it is sent to a downstream component.</td>
</tr>
<tr>
<td>CWE-78</td>
<td>Improper Neutralization of Special Elements used in an OS Command (&quot;OS Command Injection&quot;)</td>
<td>The software constructs all or part of an OS command using externally-influenced input from an upstream component, but it does not neutralize or incorrectly neutralizes special elements that could modify the intended OS command when it is sent to a downstream component.</td>
</tr>
<tr>
<td>CWE-88</td>
<td>Argument Injection or Modification</td>
<td>The software does not sufficiently delimit the arguments being passed to a component in another control sphere, allowing alternate arguments to be provided, leading to potentially security-relevant changes.</td>
</tr>
</tbody>
</table>
Increases in the Number of CISQ Weaknesses

<table>
<thead>
<tr>
<th>Quality Attribute</th>
<th>Parent weaknesses</th>
<th>Child weaknesses</th>
<th>Previous weaknesses</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reliability</td>
<td>36</td>
<td>38</td>
<td>29</td>
</tr>
<tr>
<td>Security</td>
<td>36</td>
<td>37</td>
<td>22</td>
</tr>
<tr>
<td>Performance</td>
<td>16</td>
<td>3</td>
<td>15</td>
</tr>
<tr>
<td>Maintainability</td>
<td>30</td>
<td>1</td>
<td>20</td>
</tr>
<tr>
<td>Totals</td>
<td>118</td>
<td>79</td>
<td>86</td>
</tr>
</tbody>
</table>
CISQ-like Measures Predict Incidents & Costs

Correlation of Total Quality Index and log of incidents for 21 applications in a large global system integrator

$R^2 = 0.34$
Total Quality Index accounts for 1/3 of variation in incidents

Increase in Total Quality Index of 0.24 decreased corrective maintenance effort 50%
The Technical Debt Metaphor

Technical Debt — the future cost of repairing must-fix defects remaining in code at release, a component of the cost of ownership

Business Risk

Lost opportunity

Liability from debt

Technical Debt

Interest on the debt

Principal borrowed

Structural quality problems in production code

Lost opportunity — benefits that could have been achieved had resources been put on new capability rather than retiring technical debt

Liability — business costs related to outages, breaches, corrupted data, and other damaging incidents

Interest — continuing IT costs attributable to the violations causing technical debt, i.e., higher maintenance costs, greater resource usage, etc.

Principal — cost of fixing problems remaining in the code after release that must be remediated

© 2019 Consortium for IT Software Quality (CISQ)  www.it-cisq.org

Automated Technical Debt Measure

Sum of all efforts-to-fix for all weaknesses in each CISQ Structural Quality Measure

- Reliability weaknesses
- Security weaknesses
- Performance weaknesses
- Maintainability weaknesses

Sum of efforts-to-fix for all instances of each weakness

Weighted effort-to-fix for each instance of a weakness

Predict effort for corrective maintenance

Predict cost of corrective maintenance
CISQ Roadmap

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Size</strong></td>
<td>Automated Function Points</td>
<td>Automated Enhancement Points</td>
<td>Acquisition Management</td>
<td>Risk &amp; Cost Management</td>
<td>Transformation to Agile &amp; Automation</td>
<td>Model-Based Software Engineering Measures</td>
</tr>
<tr>
<td><strong>Quality</strong></td>
<td>ISO 25010</td>
<td>Automated Source Code Quality Measures</td>
<td>Reliability</td>
<td>Security</td>
<td>Performance</td>
<td>Maintainability</td>
</tr>
<tr>
<td></td>
<td>CWE</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>ISO 25023</td>
<td>Automated Technical Debt Measure</td>
<td>Certification</td>
<td>Due Diligence</td>
<td>Benchmarking</td>
<td></td>
</tr>
<tr>
<td><strong>Effort</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
CISQ Conforms/Supplements ISO 25000 standards

- ISO/IEC 25010 defines a software product quality model of 8 quality characteristics
- CISQ conforms to ISO/IEC 25010 quality characteristic definitions
- ISO/IEC 25023 defines measures, but not automatable or at the source code level
- CISQ supplements ISO/IEC 25023 with automatable source code level measures
The CISQ Security measure (and others) can be used in numerous processes of the NIST Cybersecurity Framework. Some examples:

- Empirical risk tolerance thresholds for software security
- Contractual SLAs and audits for software security
- Evaluation of software assets for security weaknesses
- Continual improvement of software security
- Periodic scans for software weaknesses
- Software security and weakness data are shared
- Security weaknesses are identified and mitigated
Application Certification Using CISQ

➢ **CISQ/OMG**
  - only assess vendor conformance
  - do not certify applications
  - program initiated in 2017

➢ **Service providers**
  - use CISQ-conformant technology
  - in a CISQ-conformant service process
  - to provide application certifications
Objective — Define quality measures based on counting severe architectural and design weaknesses that can be detected through analyzing formal models developed in Model-Based System Engineering (MBSE) languages and technologies.

Two Focii —
1. Quality of the architecture:
   - Architecture analysis might be the only way to find some weaknesses
   - Find other weaknesses earlier at the architectural level
2. Quality of the model of the architecture

Sources —
1. Architectural-level CWEs
2. Lists of architecture-level antipatterns
3. Vendor and system architect weakness lists or experiences
As a greater portion of mission, business, and safety critical functionality is committed to software-intensive systems, these systems become one of, if not the largest source of risk to enterprises and their customers. Since corporate executives are ultimately responsible for managing this risk, we establish the following principles to govern system development and deployment.

1. Engineering discipline in product and process
2. Quality assurance to risk tolerance thresholds
3. Traceable properties of system components
4. Proactive defense of the system and its data
5. Resilient and safe operations