MEMORANDUM

TO: File No. S7-01-13

FROM: Dhawal Sharma

RE: Proposed Regulation Systems Compliance and Integrity

DATE: July 12, 2013

On June 28, 2013, Commission staff had a meeting with representatives of CAST to discuss proposed Regulation SCI.

Commission staff included David Shillman, Todd Scharf, Heidi Pilpel, Elizabeth Badawy, Harrison Lou, Keith Riley, Yue Ding, and Dhawal Sharma from the Division of Trading and Markets.

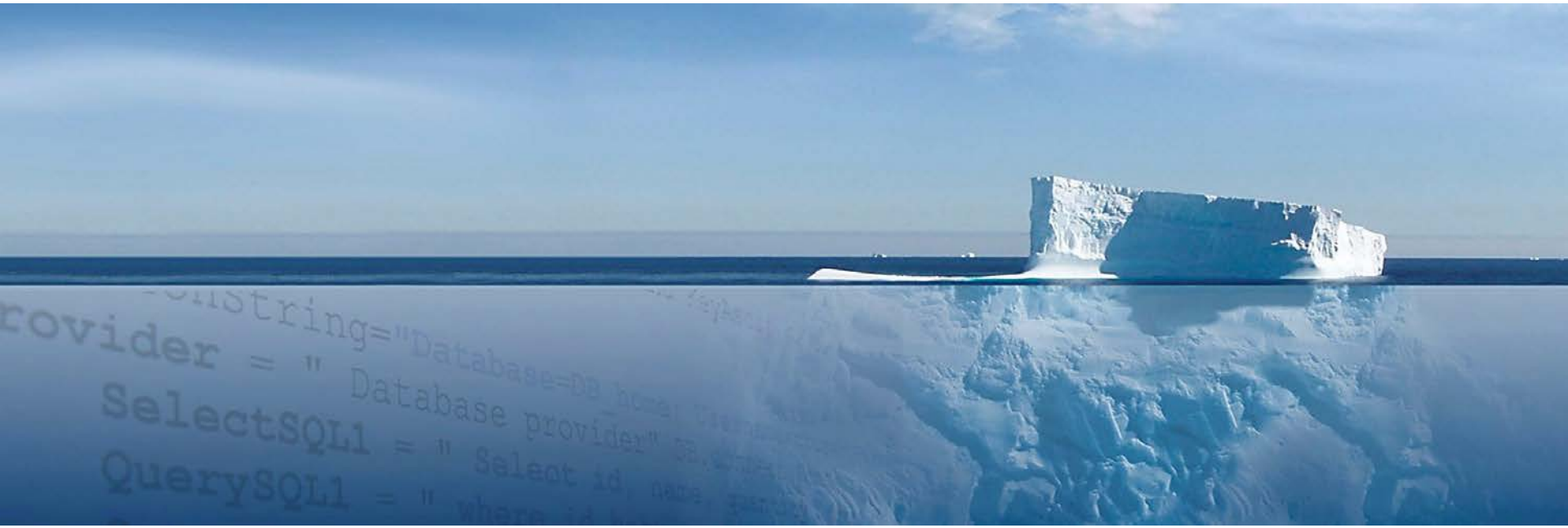
The CAST representatives at the meeting were Lev Lesokhin, Bill Curtis, Mark Jones, and Pete Pizzutillo.

The topics discussed included: (1) introduction to CAST, and context for involvement and comments; (2) CAST research on structural quality and impact on software risk; (3) current state of IT software quality in financial services industry; (4) review of other federal government policies on structural quality and assurance; (5) review of the overall approaches to managing software risk and quality; and (6) review of the evolution of standards landscape and procedures in the industry.

The attached documents were distributed by the CAST representatives and discussed at the meeting.

CAST

ACHIEVE INSIGHT. DELIVER EXCELLENCE.



Briefing to SEC – System Compliance & Integrity

Measurement of system integrity at a structural code level

June 2013

Meeting agenda

- Intro to CAST, and context for our involvement/comments
- CAST research on structural quality and impact on software risk
- Current state of IT software quality in financial services industry
- Review other Federal government policy on structural quality & assurance
- Review overall approaches to managing software risk and quality
- Review evolution of standards landscape and procedures in the industry
  - Capability Maturity Model (CMM)
  - Consortium for IT Software Quality (CISQ), ISO
  - Software Assurance standards (CWE, OWASP, etc.)
- Wrap up and discussion

CAST introduction & basis for opinion on SCI

Long term mission	Transform application development and sourcing into a management discipline through measurement and transparency
Market presence	• Established market presence in North America, Europe and India • Broadly endorsed by industry thought leaders • Strong presence at top brands in Financial Services, Public Sector and other IT-intensive industries
Research-driven focus	▪ Largest IT structural quality benchmarking database in the world ▪ Over $100 million of investment in R&D, driven by top talent in software engineering ▪ CAST Research Labs, a premier R&D facility dedicated to the science of Software Analysis & Measurement (SAM)

IDC Analyze the Future

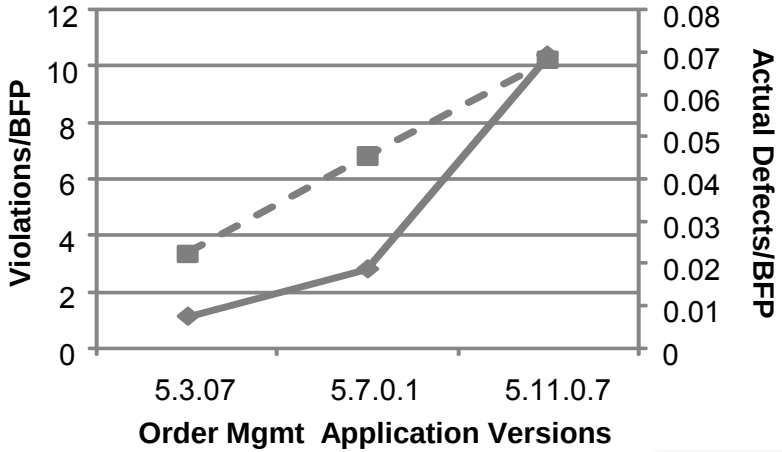
CAST is a leader in applying software quality analysis and measurement technology in the IT space.

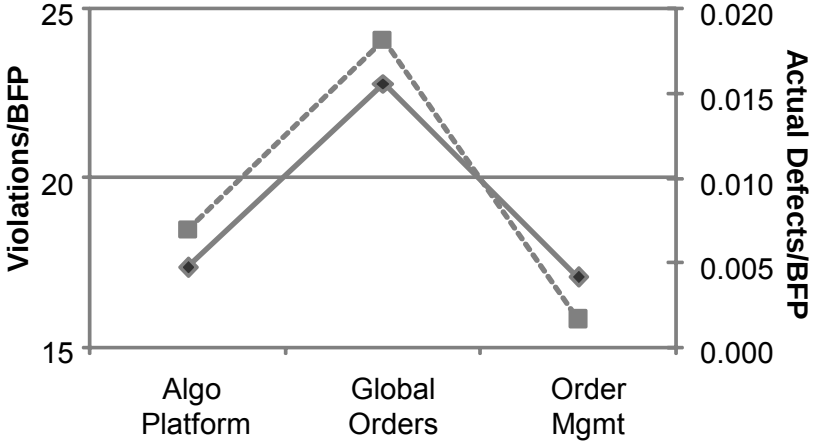
Gartner | 2011 COOL VENDOR

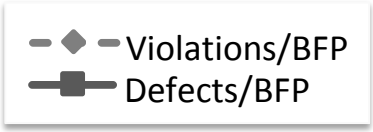
CAST metrics have become the de facto standard for measuring the quality and productivity of application services.

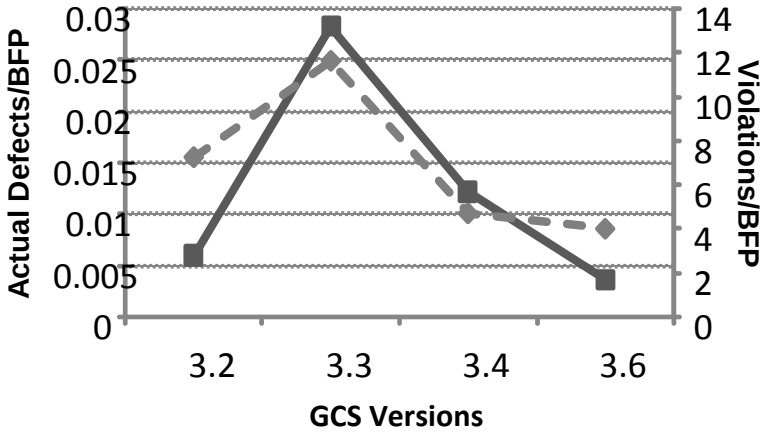
Structural issues correlate highly to SW defects

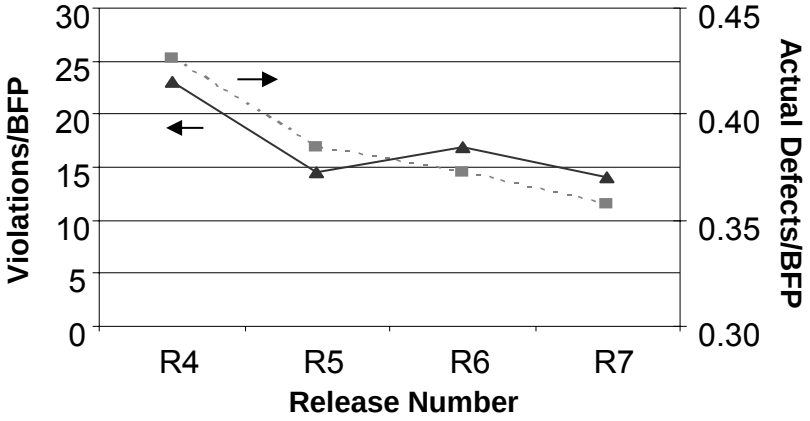
INSTITUTIONAL TRADING PLATFORMS





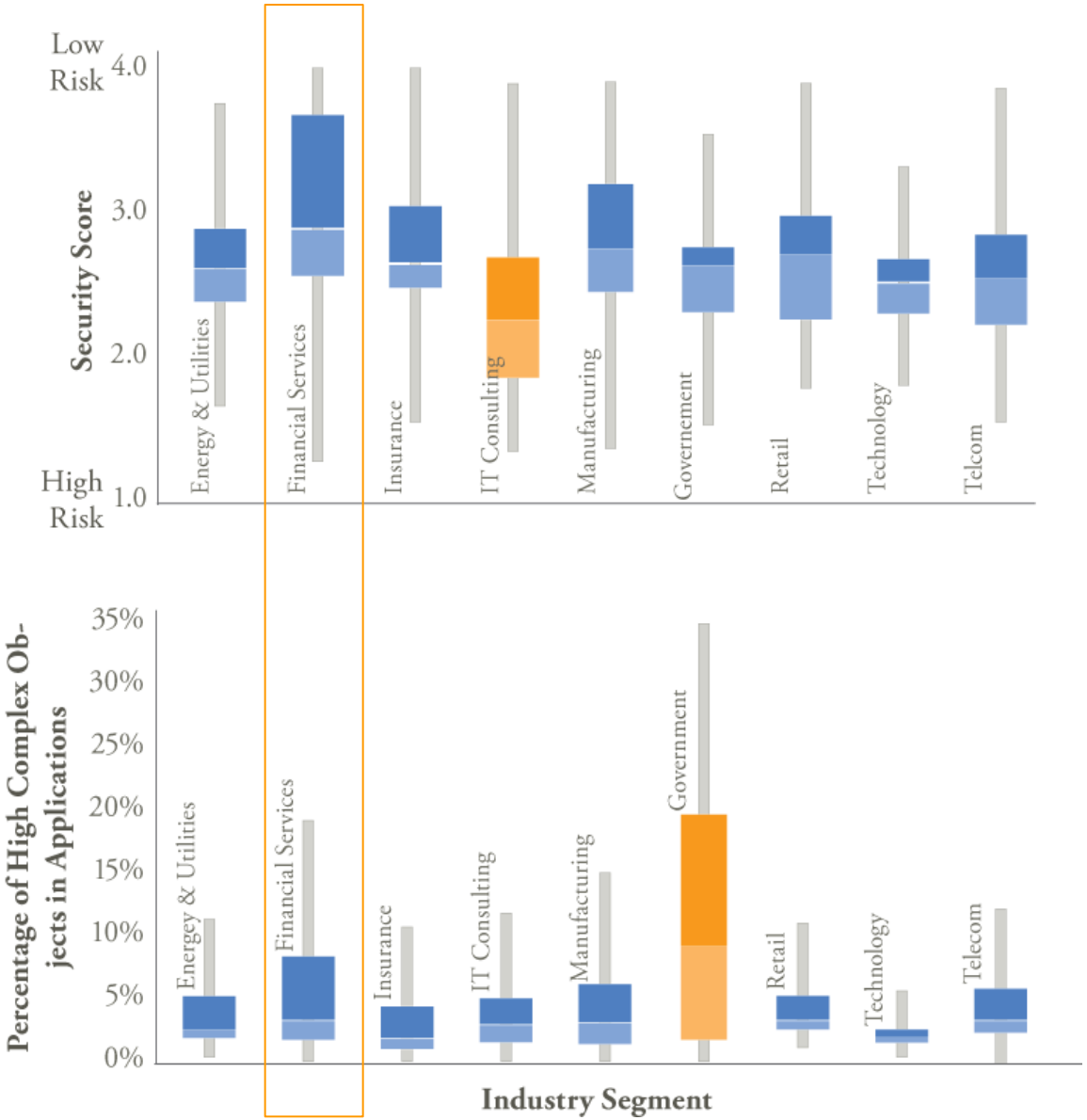




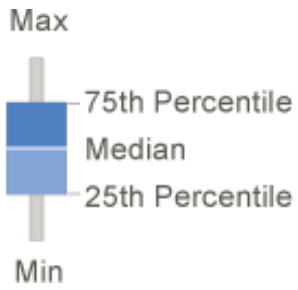


CAST

Financial services is more secure, but more complex



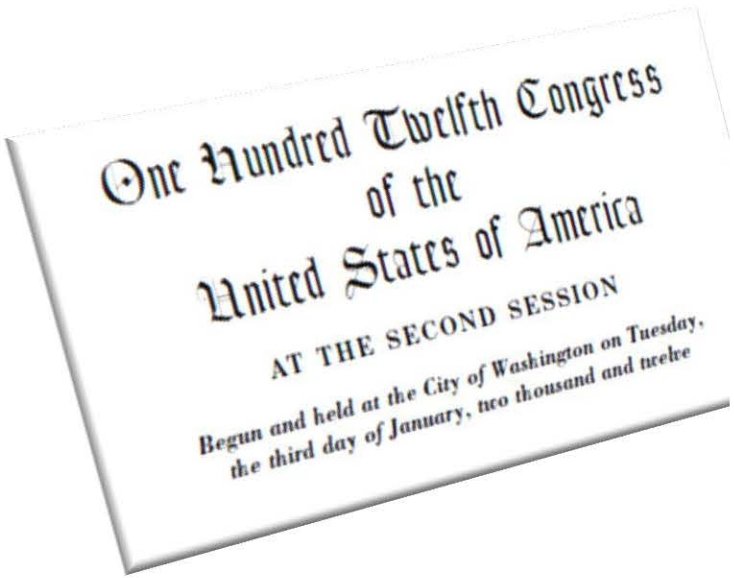
- Financial Services technology has better security than peer industries
- But, after the public sector, Financial Services has the most complex systems



Source: CAST Research Labs study – CRASH 2011; n=745 application, 365 million lines of code

CAST

NDAA Section 933 – software quality & assurance


One Hundred Twelfth Congress
of the
United States of America

AT THE SECOND SESSION

Begun and held at the City of Washington on Tuesday, the third day of January, two thousand and twelve

(a) BASELINE SOFTWARE ASSURANCE POLICY.—The Under Secretary of Defense for Acquisition, Technology, and Logistics, in coordination with the Chief Information Officer of the Department of Defense, shall develop and implement a baseline software assurance policy for the entire lifecycle of covered systems. Such policy shall be included as part of the strategy for trusted defense systems of the Department of Defense.

(b) POLICY ELEMENTS.—The baseline software assurance policy under subsection (a) shall—

(1) require use of appropriate automated vulnerability analysis tools in computer software code during the entire lifecycle of a covered system, including during development, operational testing, operations and sustainment phases, and retirement;

(2) require covered systems to identify and prioritize security vulnerabilities and, based on risk, determine appropriate remediation strategies for such security vulnerabilities;

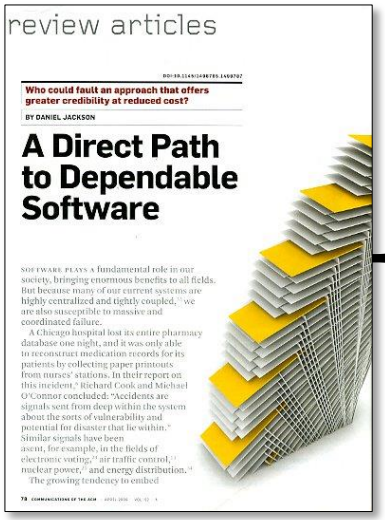
(3) ensure such remediation strategies are translated into contract requirements and evaluated during source selection;

H. R. 4310—254

(4) promote best practices and standards to achieve software security, assurance, and quality; and

(5) support competition and allow flexibility and compatibility with current or emerging software methodologies.

Testing is Not Enough


review articles

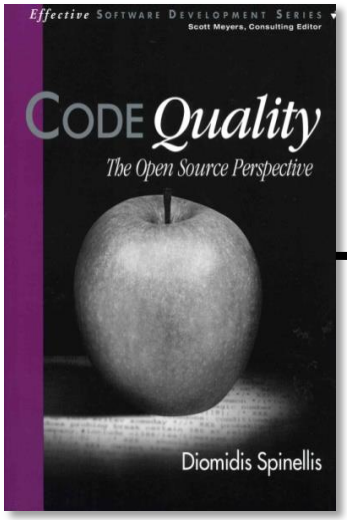
Who could fault an approach that offers greater credibility at reduced cost?

BY DANIEL JACKSON

A Direct Path to Dependable Software

"As higher levels of assurance are demanded…testing cannot deliver the level of confidence required at a reasonable cost."

"The correctness of the code is rarely the weakest link."



"…a failure to satisfy a non-functional requirement can be critical, even catastrophic…non-functional requirements are sometimes difficult to verify. We cannot write a test case to verify a system's reliability…The ability to associate code to non-functional properties can be a powerful weapon in a software engineer's arsenal."

Jackson, D. (2009). Communications of the ACM, 52 (4); Spinellis, D. (2006). Code Quality. Addison-Wesley.

CAST

Would you put untested code into operation?

If functional testing at the code unit level is inadequate for release, why would code review at unit level be adequate?

System Level (Quality Assurance)

System Testing
(functional defect removal)

SIT, Performance Tests

Structural Analysis & Control
(Non-functional Defect Removal—Reliability, Performance, Security, Maintainability)

Build and Integration

Code Unit Level (Developer)

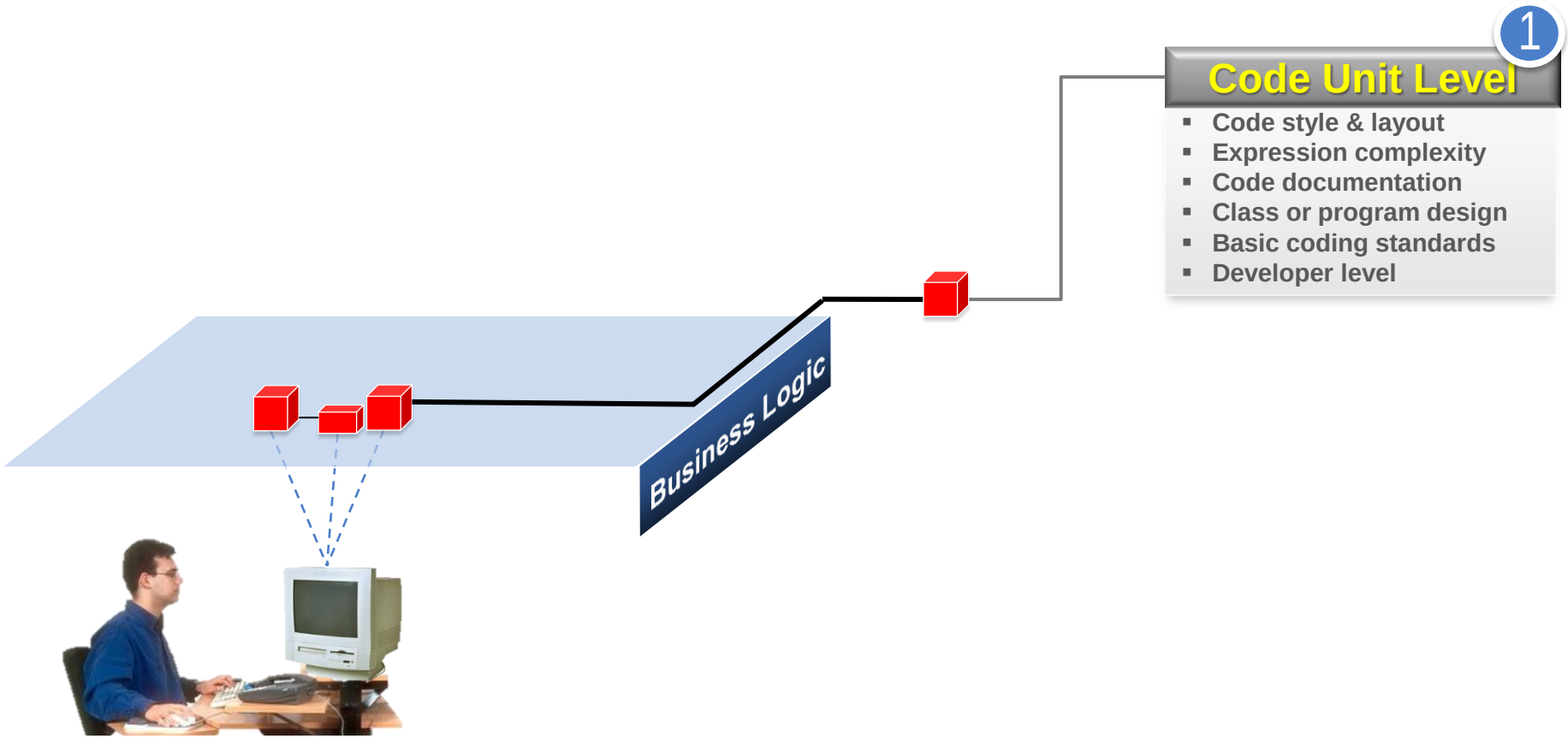
Functional Unit Tests
(code unit correctness)

IDE Unit Testing

Coding Best Practices
(readability, code unit reliability)

IDE Static Analysis

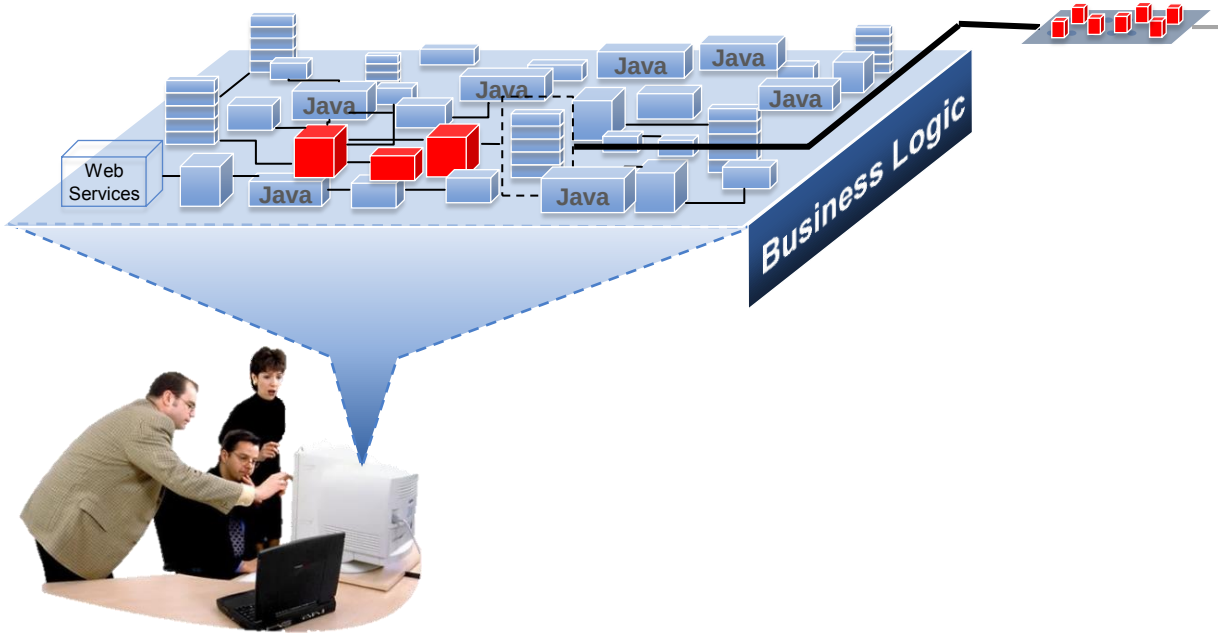
Code Unit Level — Can be performed by developer



IDE Static Analysis tools

Developer level code unit analysis

Technology Level — Requires tools and program focus



Single language commercial static analysis tools

Quality Assurance

1

Code Unit Level

- Code style & layout
- Expression complexity
- Code documentation
- Class or program design
- Basic coding standards
- Developer level

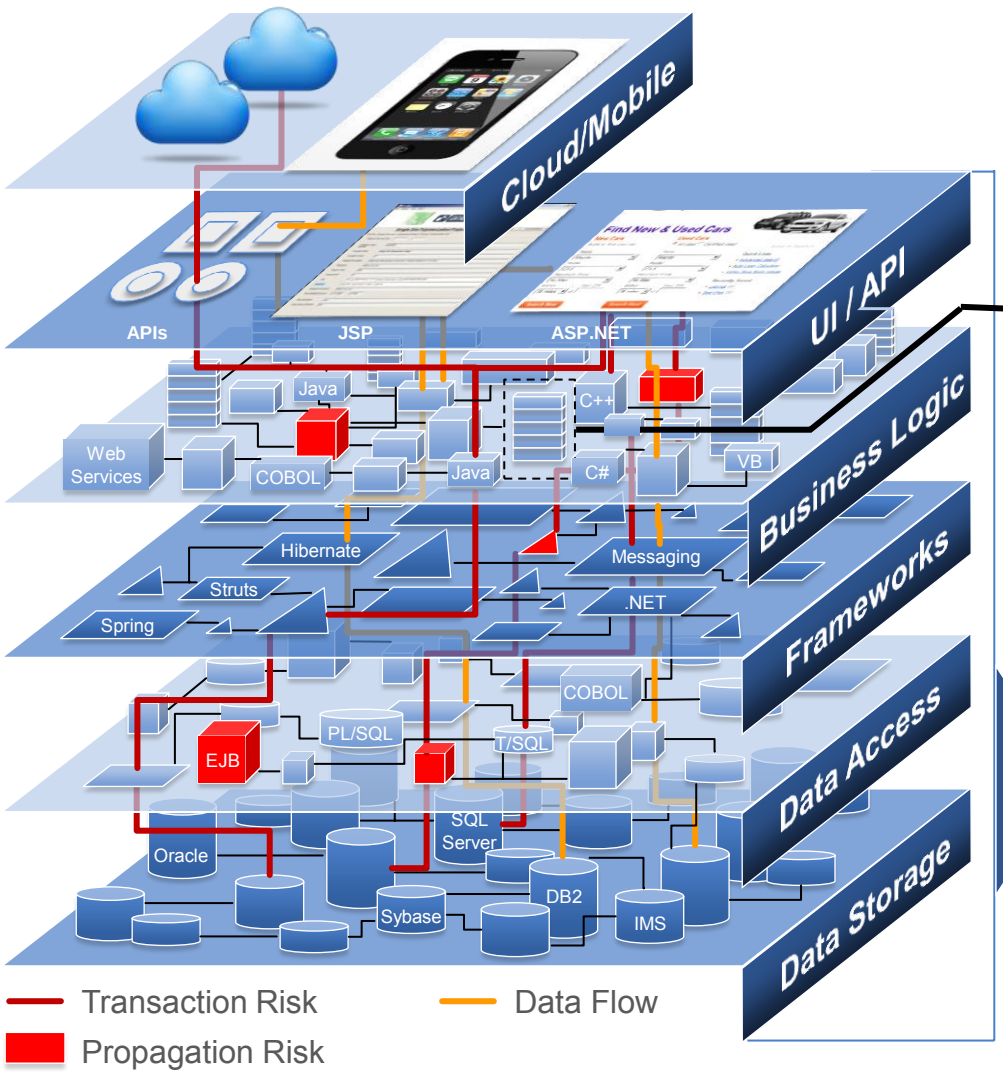
2

Technology Level

- Single language/technology layer
- Intra-technology architecture
- Intra-layer dependencies
- Design & structure
- Inter-program invocation
- Security vulnerabilities
- Development team level

CAST

System Level — Requires holistic analysis, across teams



1 Code 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
- Design & structure
- Inter-program invocation
- Security vulnerabilities
- Development team level

3 Application Stack 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

CAST

Standards in software measurement & assurance

- **Software product quality**
  - ISO 9126, CISQ, ISO 25000
- **Software sizing**
  - IFPUG, OMG/CISQ, SEI
- **Large system engineering**
  - SEI

- **CWE, CVE, MAEC, Software Assurance Forum**
  - MITRE, NIST, DHS
- **SANS Institute**
- **COBIT/ISACA, BSIMM**

CAST

CMMI & Application Quality Engineering

Application Quality Engineering supplements CMMI to better control risk in applications

	Focus	*Similar program*
CMMI	**Process** improvement	*Six Sigma*
AQE	**Product** improvement	*Design for Six Sigma*

CISQ

CONSORTIUM FOR IT SOFTWARE QUALITY

Standard Metrics to Manage Software Risk

Dr. Bill Curtis
Director, CISQ

June 28, 2013

Software Engineering Institute | Carnegie Mellon

OMG OBJECT MANAGEMENT GROUP

Consortium for IT Software Quality

Carnegie Mellon Software Engineering Institute

OBJECT MANAGEMENT GROUP

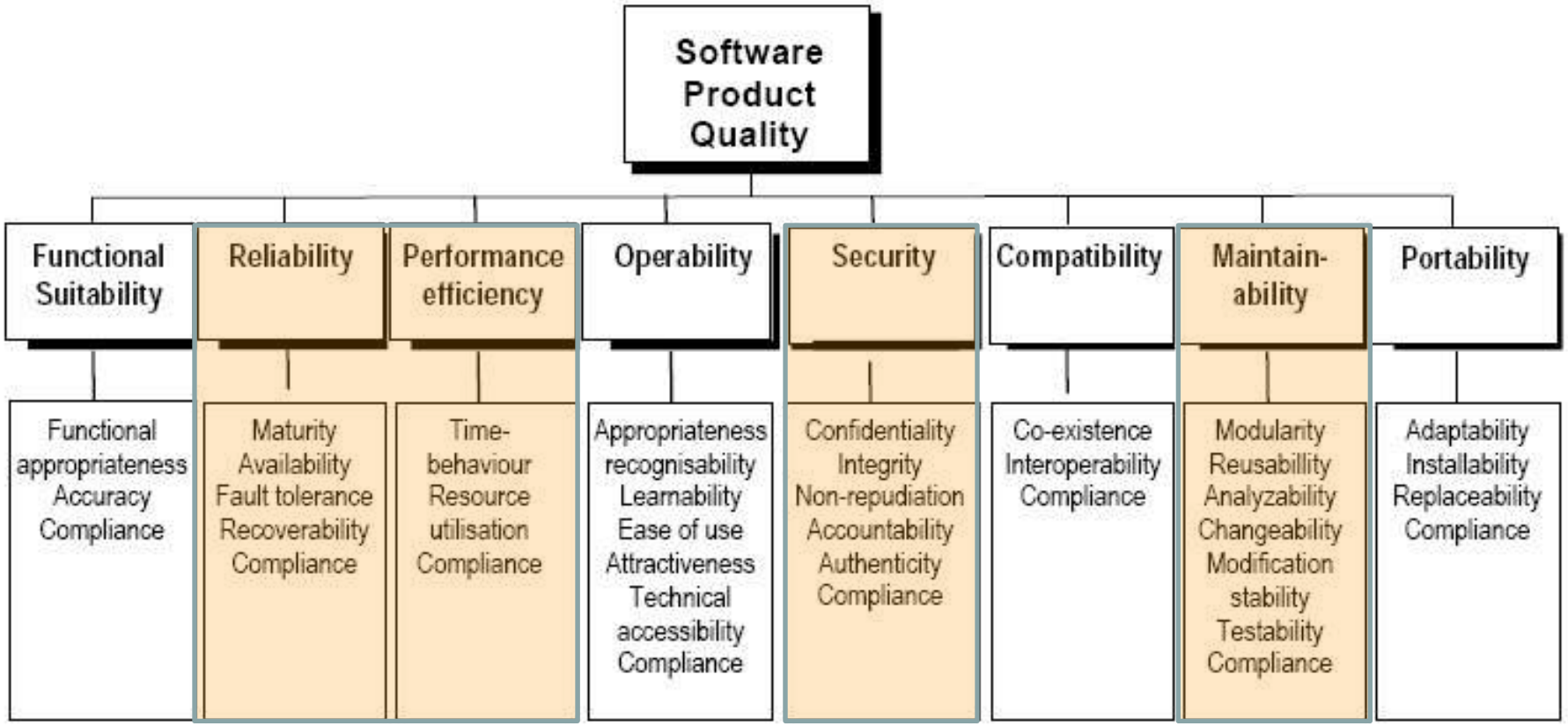
Co-sponsorship

IT Executives → **CISQ** ← **Technical experts**

www.it-cisq.org

CISQ Measures & ISO 25010

- **Starting point for CISQ work**
  - **Defines quality characteristics and sub-characteristics**
  - **CISQ to define quality attributes and measurable elements**



CISQ Security Measure

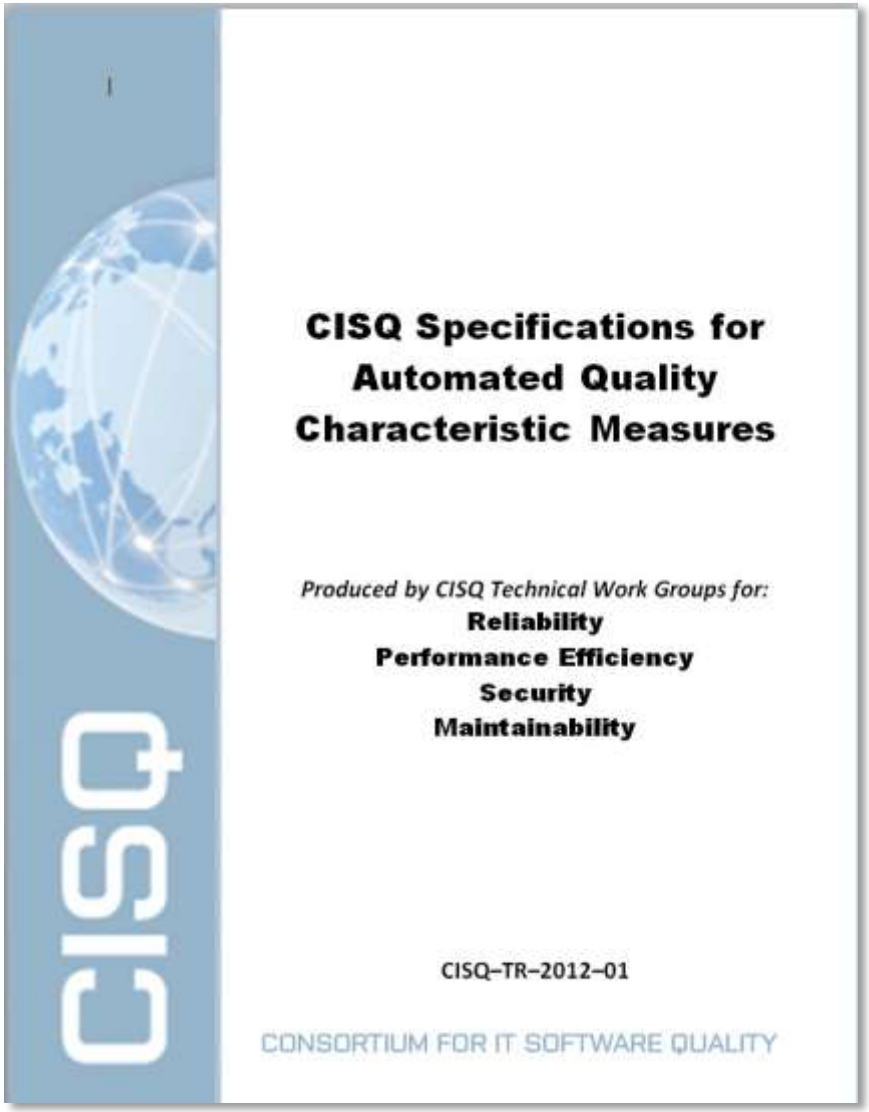
Team Lead

Robert Martin

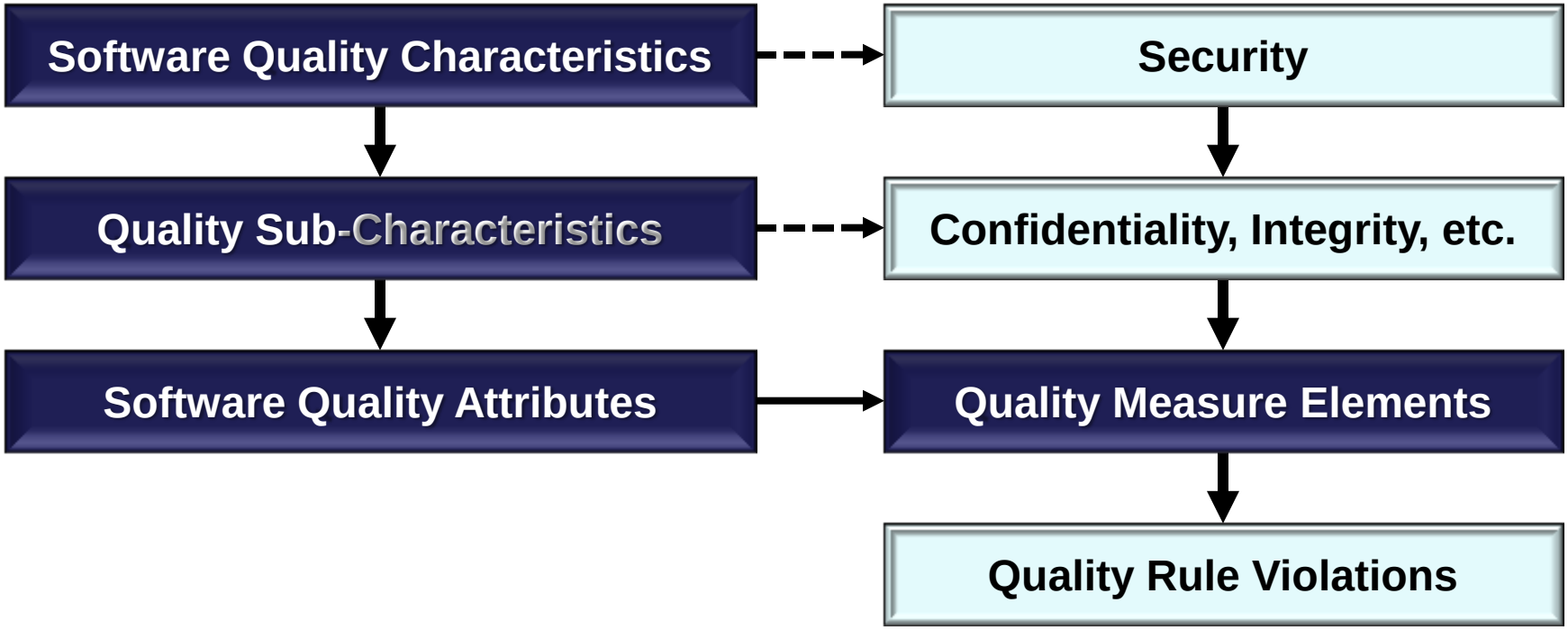
MITRE

Objective

Develop automated source code measures that predict the vulnerability of source code to external attack. Base measure on the Top 25 in the Common Weakness Enumeration



Quality Characteristic Hierarchy



- SQL injection
- Cross-site scripting
- Buffer overflows
- Functional injection

Example Quality Measure Specs

Reliability

Quality Issue	Quality Rule	Quality Measure Element
Issue 1: Inconsistent or incomplete handling of errors and exceptions leads to inaccurate identification and inadequate response to errors	**Rule 1:** Exception handling blocks such as Catch and Finally blocks must not be empty.	**Measure 1:** # of exception handling blocks such as Catch and Finally blocks that are empty **Measure 2:** # of generic exceptions thrown and caught
	Rule 2: Methods, procedures and functions doing Insert, Update, Delete, Create Table or Select must include error management (check of database error variables or exception handling).	**Measure 3:** # of functions doing Insert, Update, Delete, Create Table, and Select that do not include error management capabilities
Issue 2: Some coding weaknesses result in unexpected and faulty behaviors	**Rule 3:** Classes that implement a serializable interface must also implement a serializable method and subfields within the object that are serializable.	**Measure 4:** # of classes that implement a serializable interface must also implement a serializable method **Measure 5:** # of classes that have

Security

Issue	Quality Rule	Quality Measure Element
CWE-79: Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')	**Rule 1:** Use a vetted library or framework that does not allow this weakness to occur or provides constructs that make this weakness easier to avoid, such as Microsoft's Anti-XSS library, the OWASP ESAPI Encoding module, and Apache Wicket.	**Measure 1:** # of instances where output is not using library for neutralization

www.it-cisq.org

1. Join CISQ
2. Contribute to the blog
3. Use CISQ standards
4. Attend CISQ seminars
   - Berlin, June 19
   - NJ, Sept. 25
   - SF, Dec. 11
5. Initiate measurement
6. Improve continually
7. Build great software

OBJECT MANAGEMENT GROUP