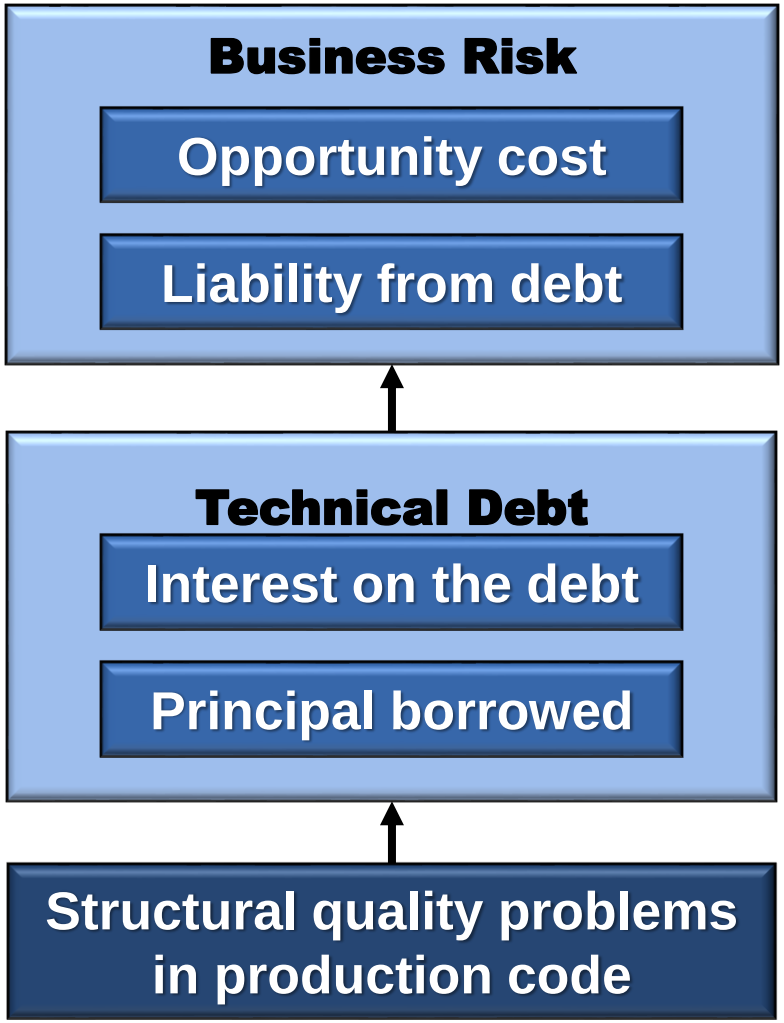
Measuring and Managing Technical Debt

Dr. Bill Curtis
SVP & Chief Scientist, CAST Research Labs
Director, Consortium for IT Software Quality

The Technical Debt Metaphor

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



Opportunity cost—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, etc.

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

Uses of Technical Debt Metaphor

Calculating Cost Of Ownership

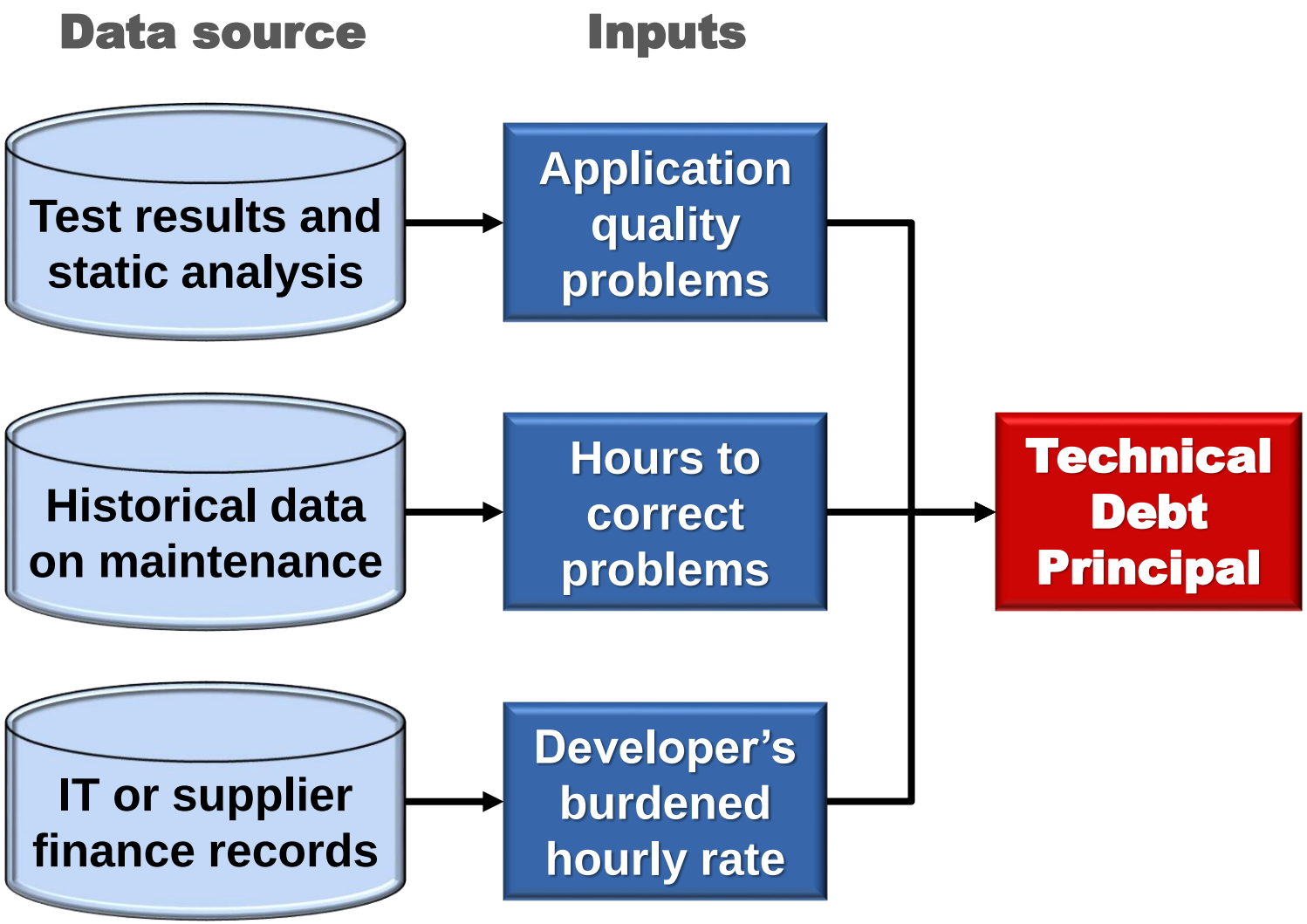
Assessing Business Risk

Estimate of Technical Debt

Explaining IT Cost of Quality

Managing Portfolio Quality

Inputs for Estimating Principal



Analyzing Structural Quality at System Level

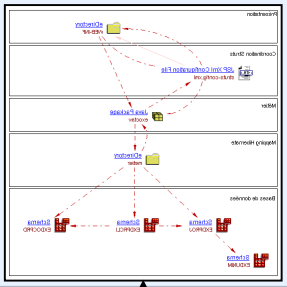
Language Parsers | **Application Analysis** | **Detected Violations** | **Quality Measurements**

- Oracle PL/SQL
- Sybase T-SQL
- SQL Server T-SQL
- IBM SQL/PSM
- C, C++, C#
- Pro C
- Cobol
- CICS
- Visual Basic
- VB.Net
- ASP.Net
- Java, J2EE
- JSP
- XML
- HTML
- Javascript
- VBScript
- PHP
- PowerBuilder
- Oracle Forms
- PeopleSoft
- SAP ABAP, Netweaver
- Tibco
- Business Objects
- Universal Analyzer for other languages

CAST Application Intelligence Platform

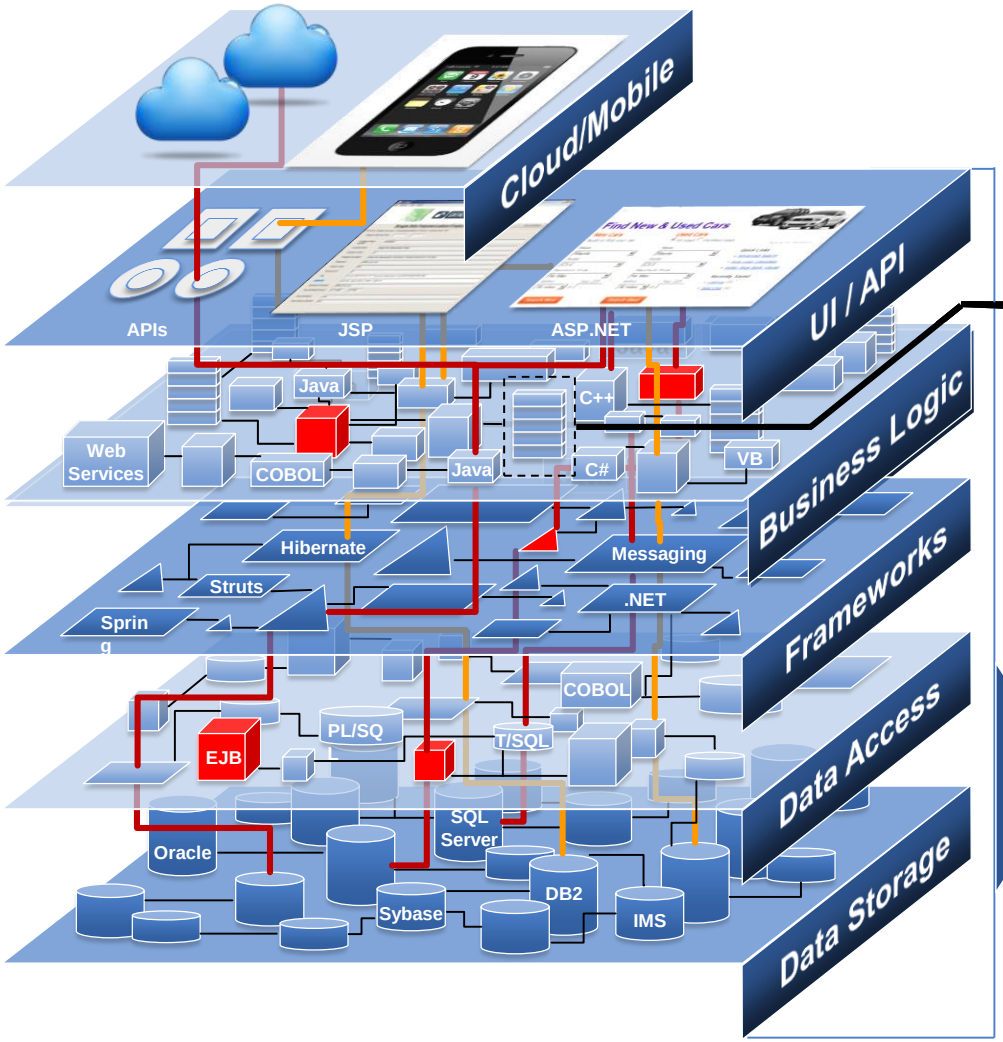
Evaluation of 1200+ coding & architectural rules

Application meta-data



Detected Violations	Quality Measurements
Expensive operation in loop Static vs. pooled connections Complex query on big table Large indices on big table	Performance
Empty CATCH block Uncontrolled data access Poor memory management Opened resource not closed	Robustness
SQL injection Cross-site scripting Buffer overflow Uncontrolled format string	Security
Unstructured code Misuse of inheritance Lack of comments Violated naming convention	Transferability
Highly coupled component Duplicated code Index modified in loop High cyclomatic complexity	Changeability

Technical Debt Is a System-wide Issue



1 Code Unit Level

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

2 Technology Level

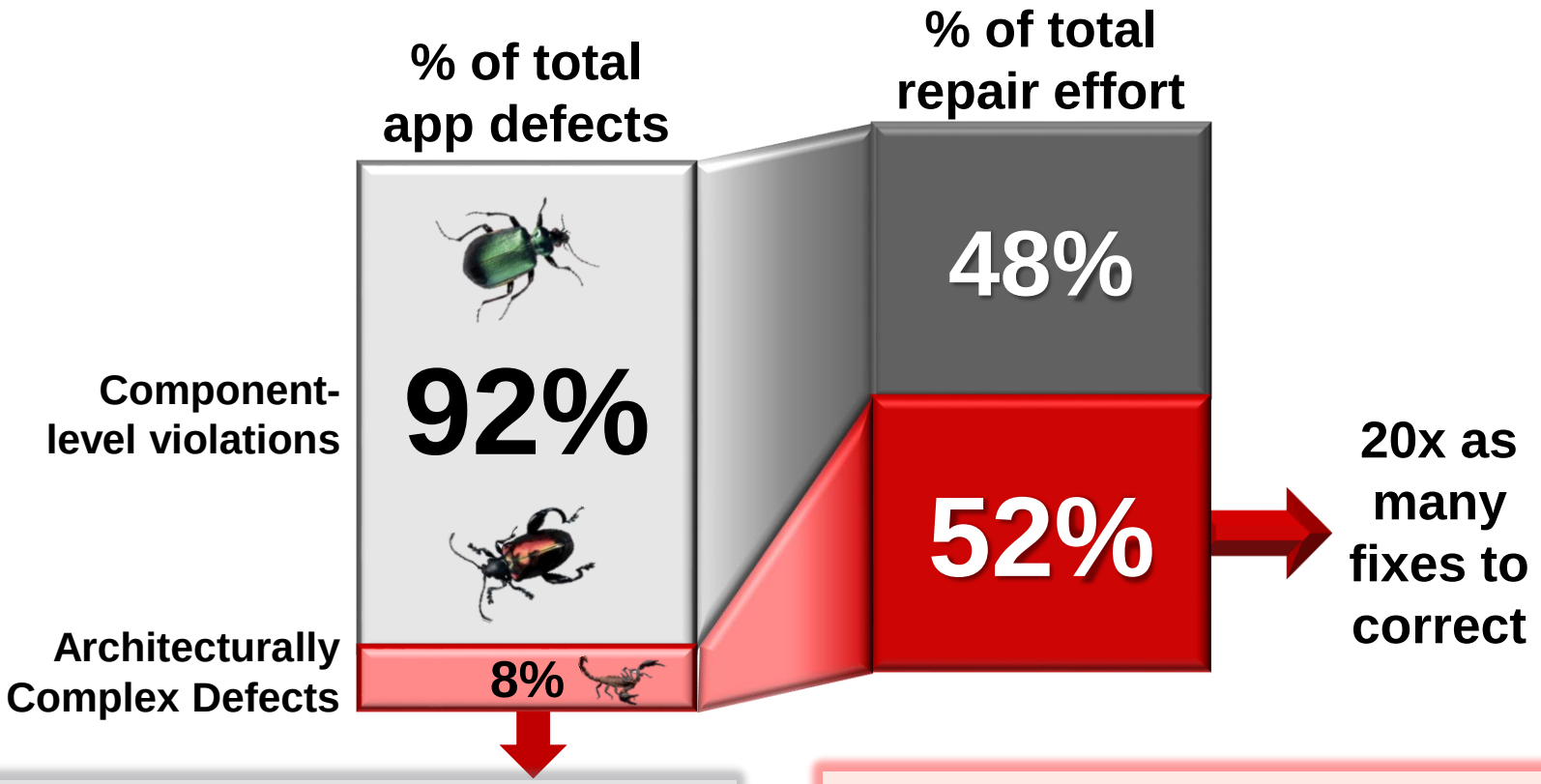
- Single language/technology layer
- Intra-technology architecture
- Intra-layer dependencies
- Design & structure
- Inter-program invocation
- Security vulnerabilities
- Development team level

3 Application 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

Architecturally Complex Defects

Architecturally Complex Defect	A structural flaw involving interactions among multiple components that reside in different application layers

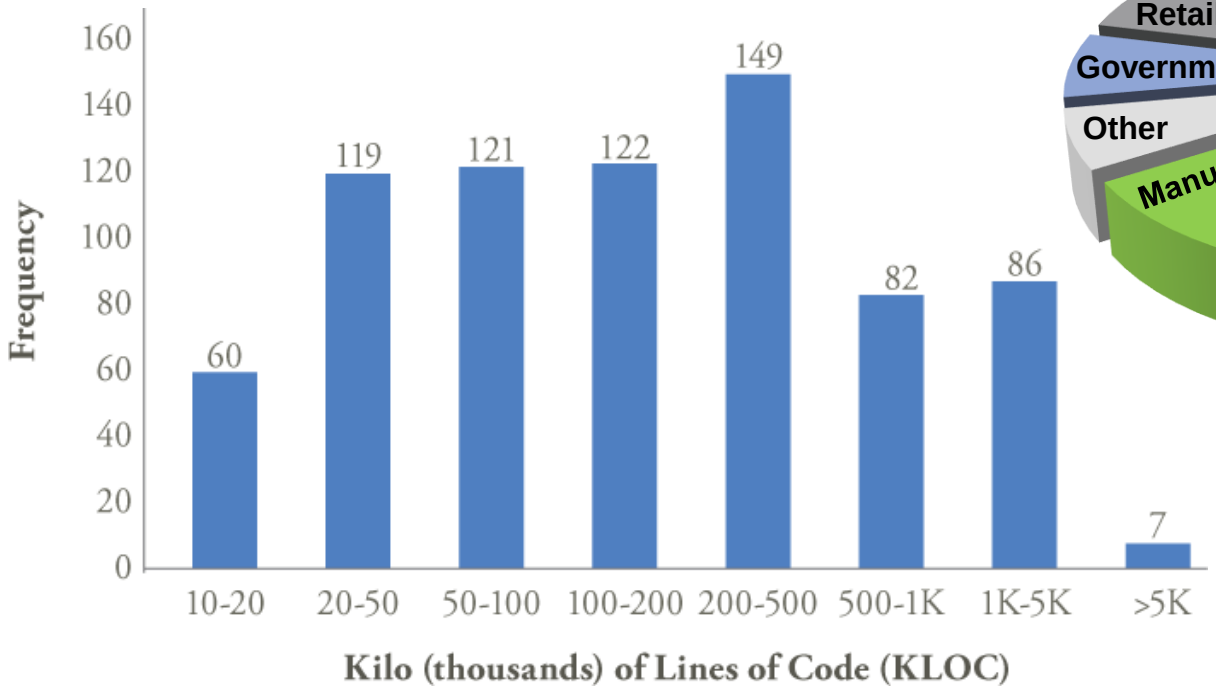


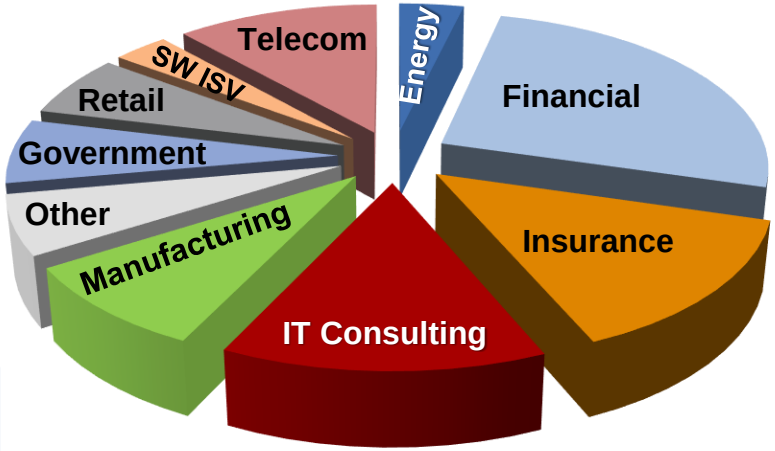
80% of architecturally complex defects touch an Architectural Hotspot—a badly designed component causing problems

Architectural hotspots provide a roadmap for remediating the worst risk, rework, and cost drivers

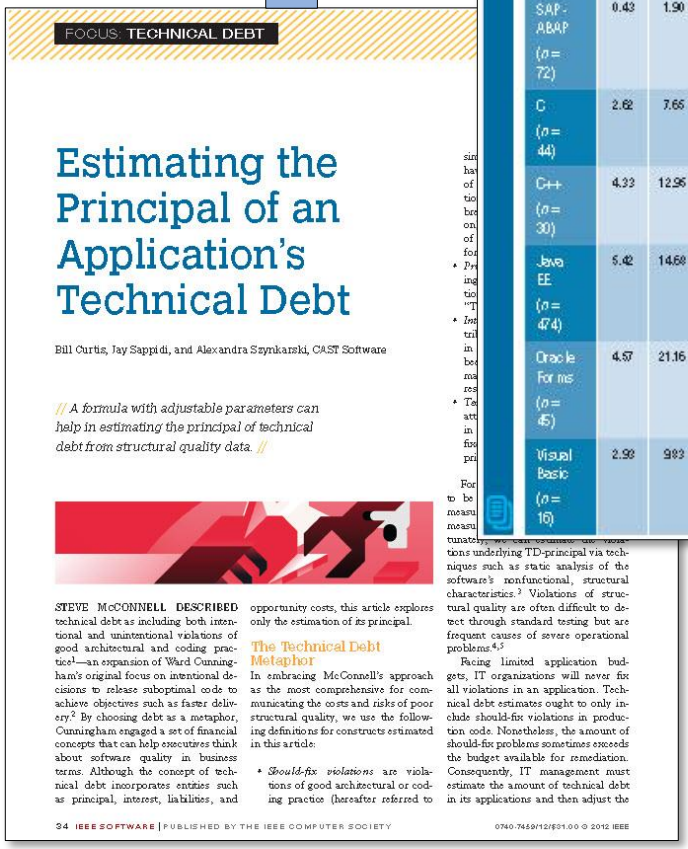
Appmarq Repository

- **Industry-leading repository on structural quality**
  - **745 Applications**
  - **160 Companies, 14 Countries**
  - **321,259,160 Lines of Code; 59,511,706 Violations**





Estimating Technical Debt


FOCUS: TECHNICAL DEBT

Estimating the Principal of an Application's Technical Debt

Bill Curtis, Jay Sappidi, and Alexandra Szynkarski, CAST Software

A formula with adjustable parameters can help in estimating the principal of technical debt from structural quality data.

STEVE McCONNELL DESCRIBED technical debt as including both intentional and unintentional violations of good architectural and coding practice[1]—an expansion of Ward Cunningham's original focus on intentional decisions to release suboptimal code to achieve objectives such as faster delivery.[2] By choosing debt as a metaphor, Cunningham engaged a set of financial concepts that can help executives think about software quality in business terms. Although the concept of technical debt incorporates entities such as principal, interest, liabilities, and opportunity costs, this article explores only the estimation of *its* principal.

The Technical Debt Metaphor

In embracing McConnell's approach as the most comprehensive for communicating the costs and risks of poor structural quality, we use the following definitions for constructs estimated in this article:

- *Should-fix violations* are violations of good architectural or coding practice (hereafter referred to

34 IEEE SOFTWARE | PUBLISHED BY THE IEEE COMPUTER SOCIETY

0740-7459/12/$31.00 © 2012 IEEE

TABLE 2

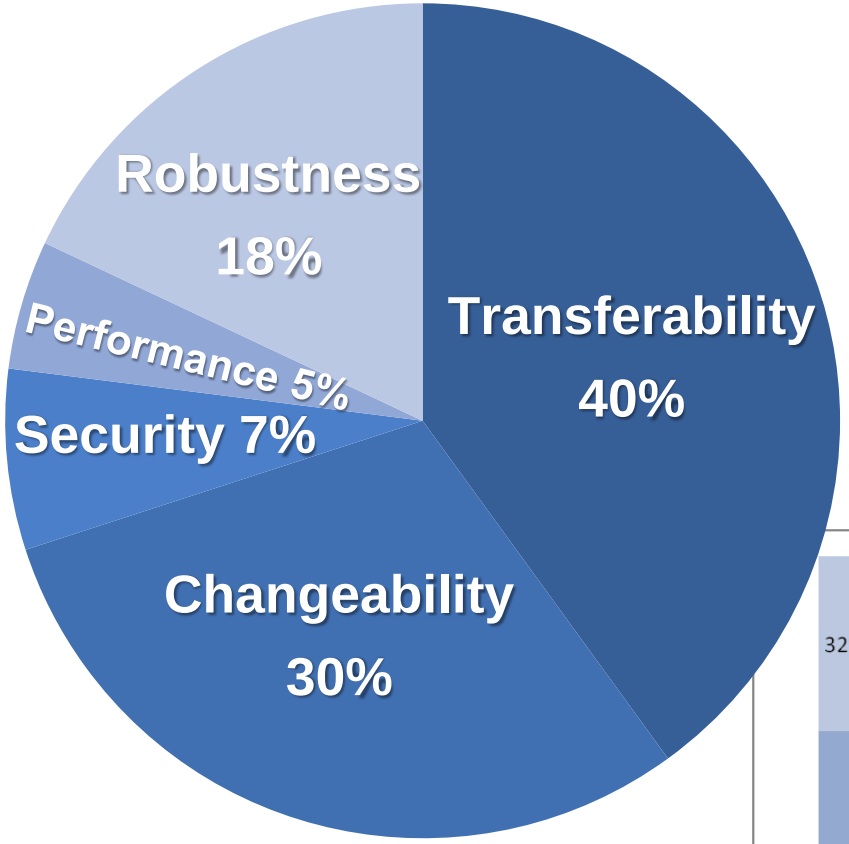
Estimated US dollars per LOC of TD-principal by language.*

	Mean			Median			Minimum			25th & 75th quartiles			Maximum		
	Est. 1	Est. 2	Est. 3	Est. 1	Est. 2	Est. 3	Est. 1	Est. 2	Est. 3	Est. 1	Est. 2	Est. 3	Est. 1	Est. 2	Est. 3
All apps (n = 700)**	3.61	10.26	15.62	2.79	7.94	11.77	0.06	0.01	0.21	1.13 5.25	3.49 14.45	5.91 19.28	38.08	132.74	278.00
.NET (n = 63)	3.09	12.29	28.34	2.37	10.20	22.32	0.96	0.49	1.18	0.84 4.58	3.35 19.06	8.02 42.01	16.52	73.00	175.63
SAP-ABAP (n = 72)	0.43	1.90	4.29	0.41	1.73	3.79	0.05	0.20	0.41	0.27 0.57	1.20 2.50	2.47 5.85	1.42	6.89	16.31
C (n = 44)	2.62	7.65	17.12	2.18	6.46	14.62	0.02	0.01	0.33	0.93 3.18	2.93 9.73	4.36 21.69	12.82	31.89	75.64
C++ (n = 30)	4.33	12.96	26.77	2.41	7.83	14.52	0.02	0.01	0.06	1.55 4.41	4.51 10.53	8.90 22.25	38.08	132.91	278.00
Java EE (n = 474)	5.42	14.68	19.82	5.13	13.66	16.18	0.07	0.23	0.50	2.40 7.48	8.19 18.52	11.94 21.33	49.72	253.03	608.68
Oracle Forms (n = 45)	4.57	21.16	49.52	1.12	3.97	7.59	0.49	1.13	1.19	0.59 5.92	3.24 27.88	5.82 66.70	20.23	151.92	366.65
Visual Basic (n = 16)	2.99	9.83	18.91	2.58	8.37	15.29	0.68	2.77	4.01	1.16 3.20	3.45 11.21	6.10 20.69	12.14	45.01	92.59

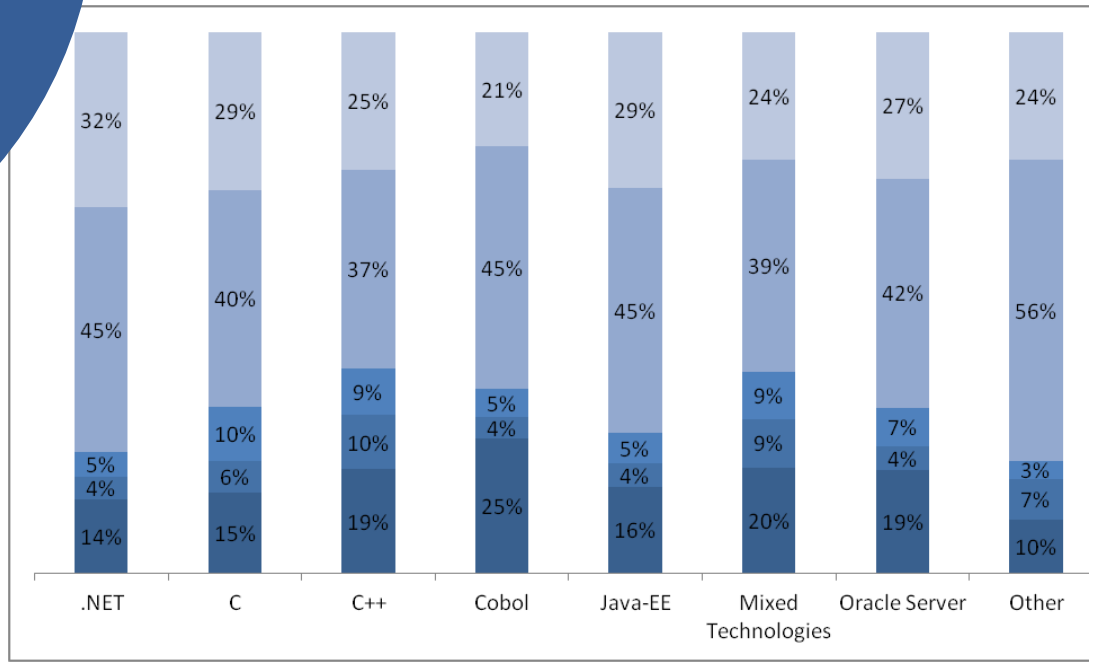
Conservative estimate: $3.61 per LOC

"Even when measured with a conservative formula, the amount of technical debt in most business applications is formidable… estimates of [technical debt] can be a powerful tool to aid management in understanding and controlling IT costs and risks."

Technical Debt by Software Quality Attribute



- **70% of Technical Debt is in IT Cost (Transferability, Changeability)**
- **30% of Technical Debt is in Business Risk (Robustness, Performance, Security)**
- **Proportions are generally consistent across technologies**



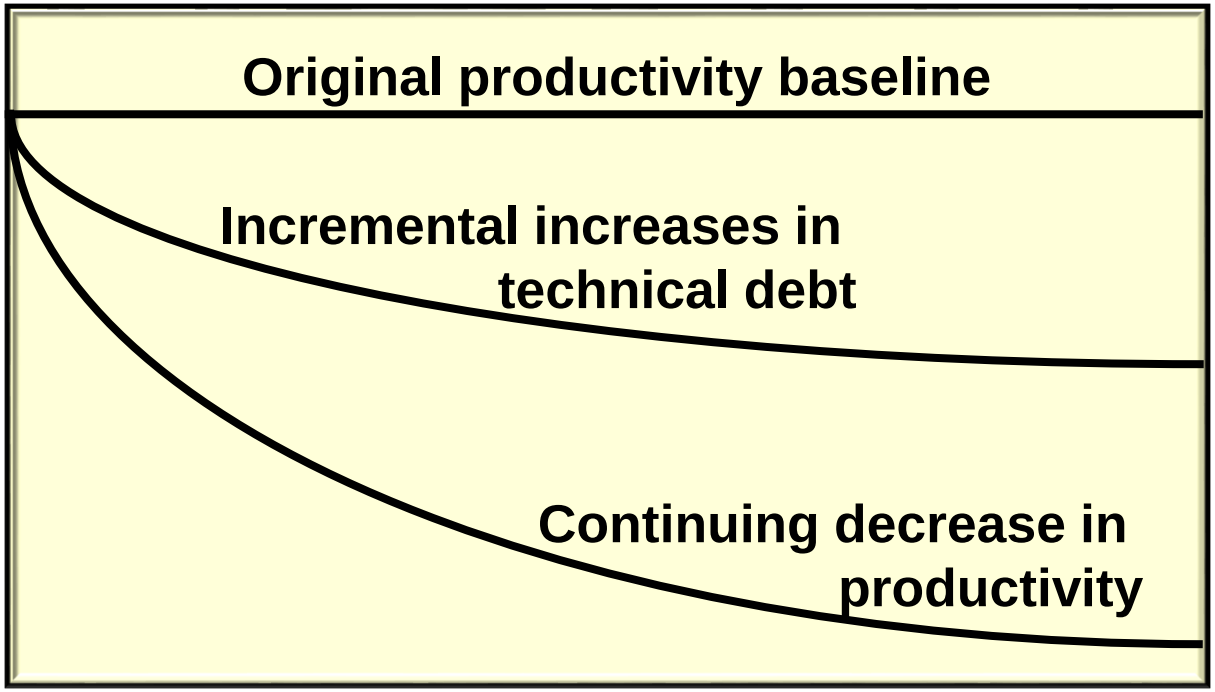
Rethinking Productivity Measurement

$$\text{Release Productivity} = \frac{\text{Volume of code developed, modified, or deleted}}{\text{Total effort expended on the release}}$$

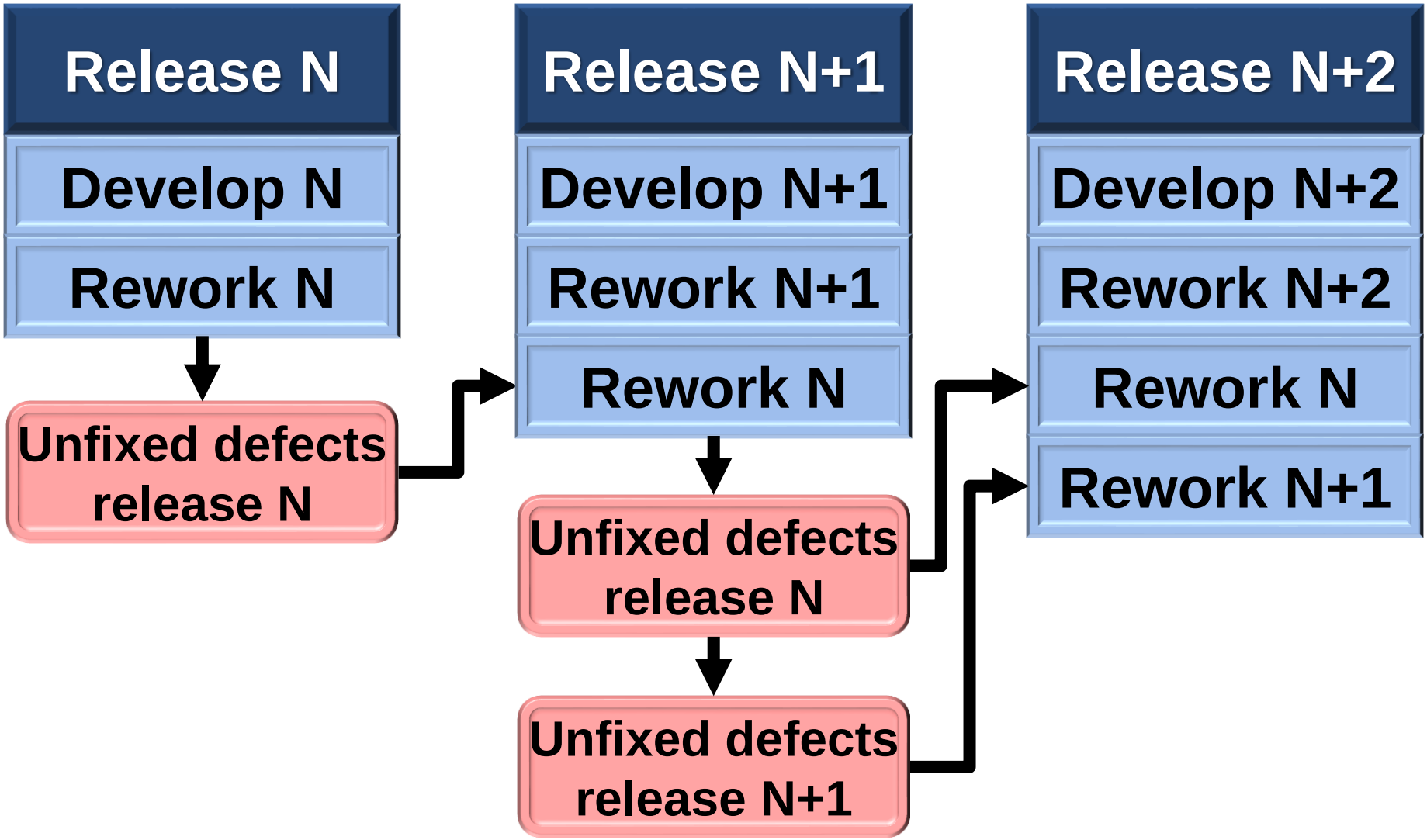
Productivity baseline —

a value in a monotonically declining function that compares the amount of product produced to the effort required to produce it

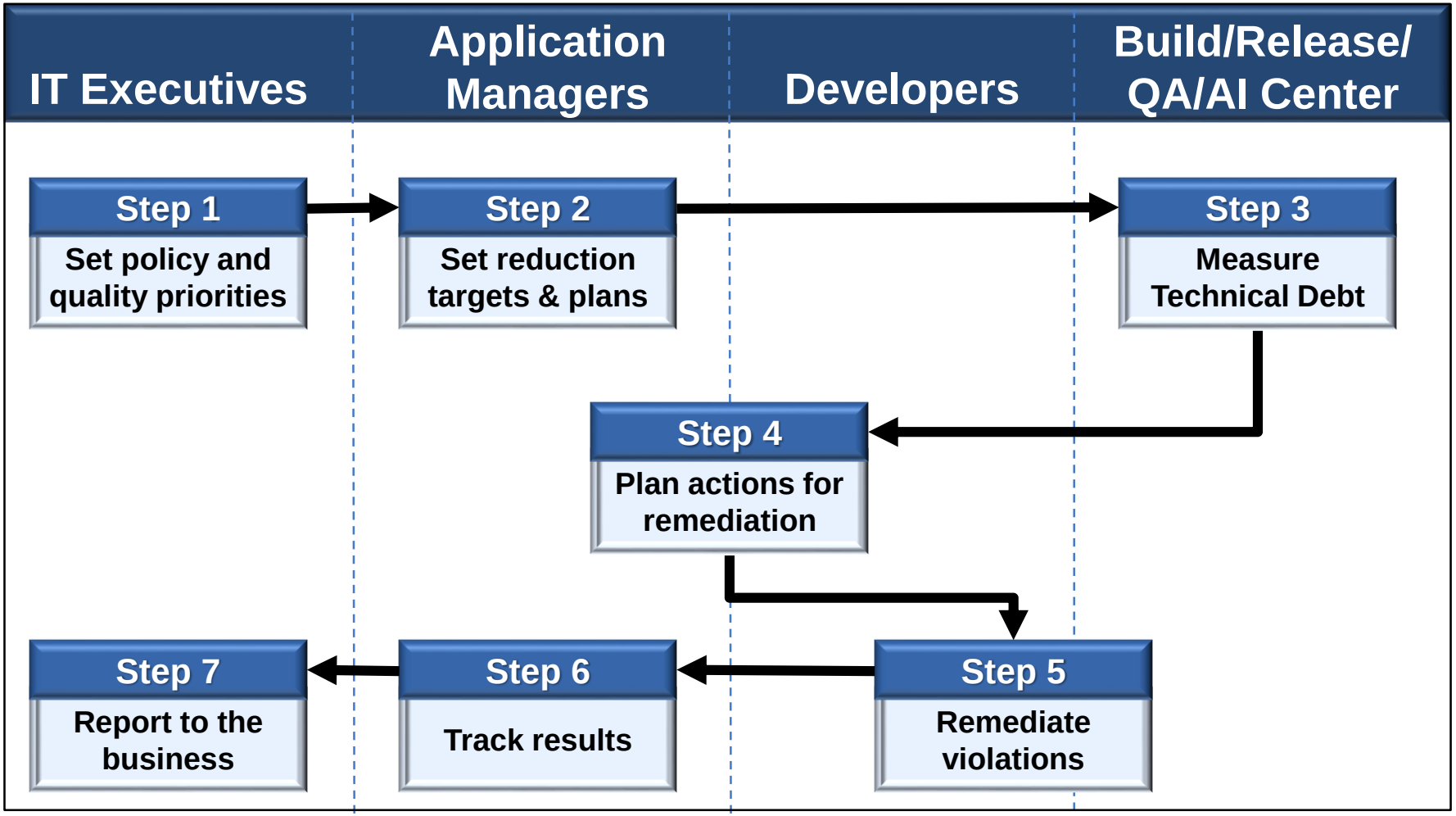
… unless you take action



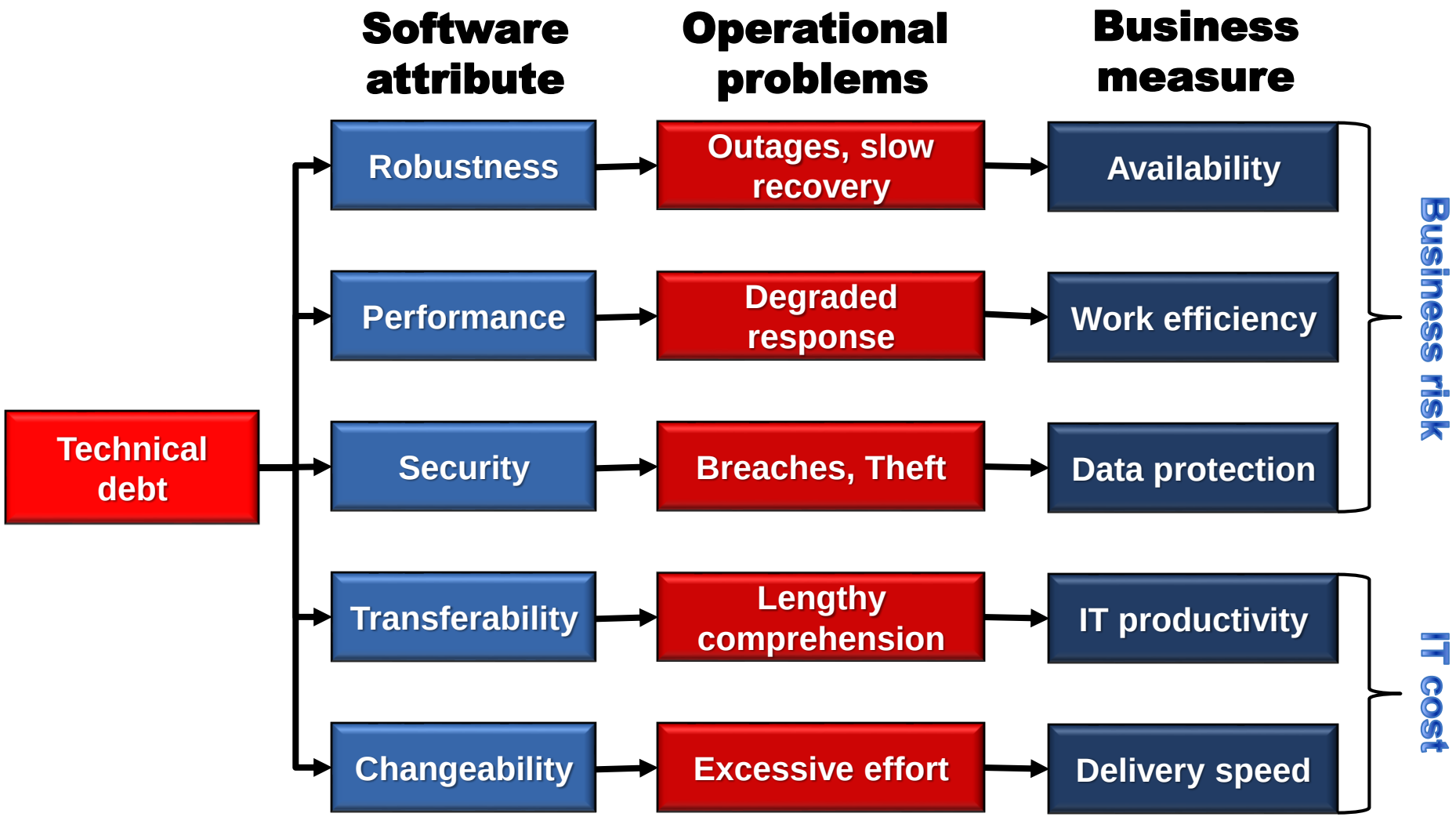
Technical Debt = Carry-forward Rework



Manage Technical Debt to Manage Productivity



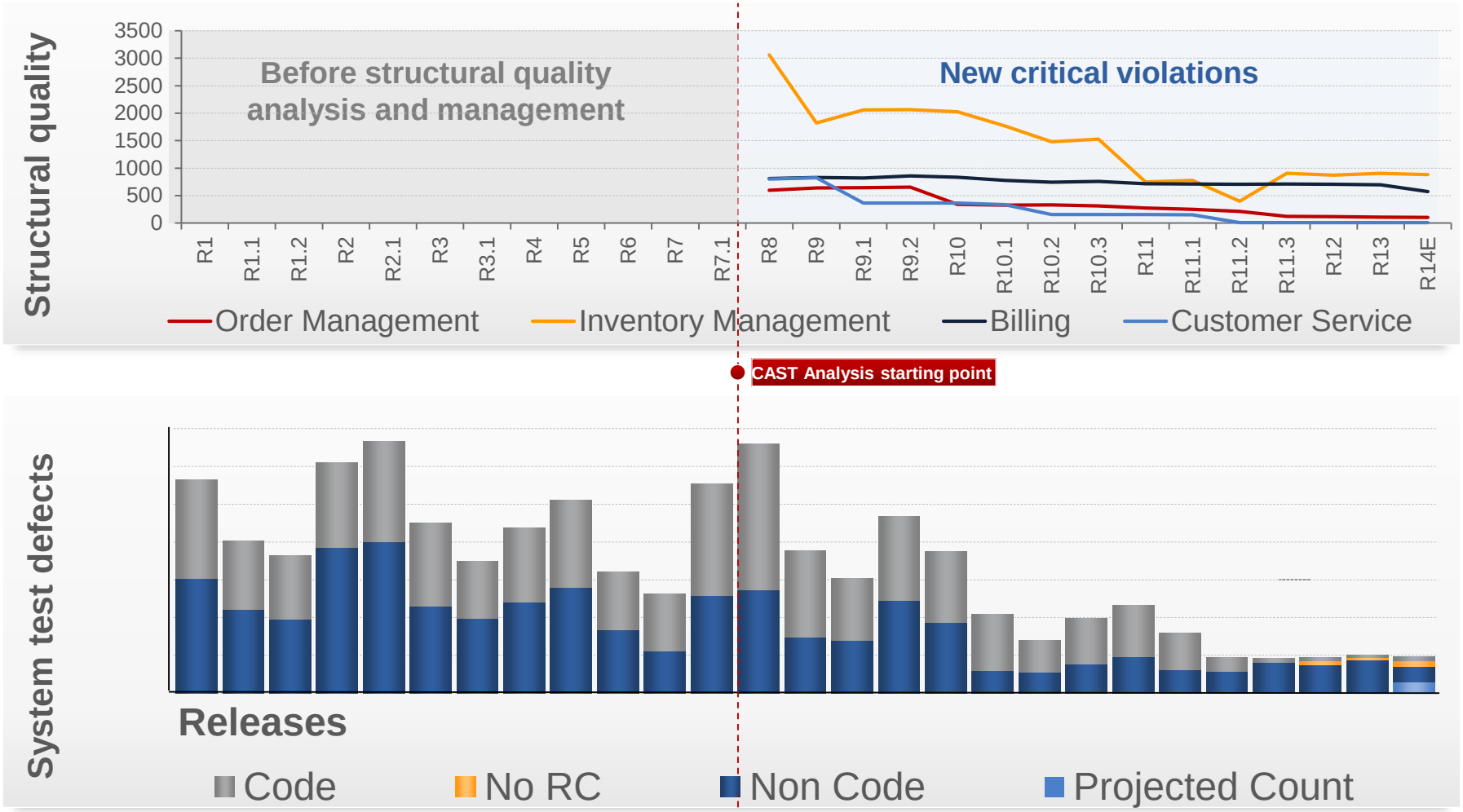
Translating Tech Debt to Business Measures



Managing Structural Quality in Telecom

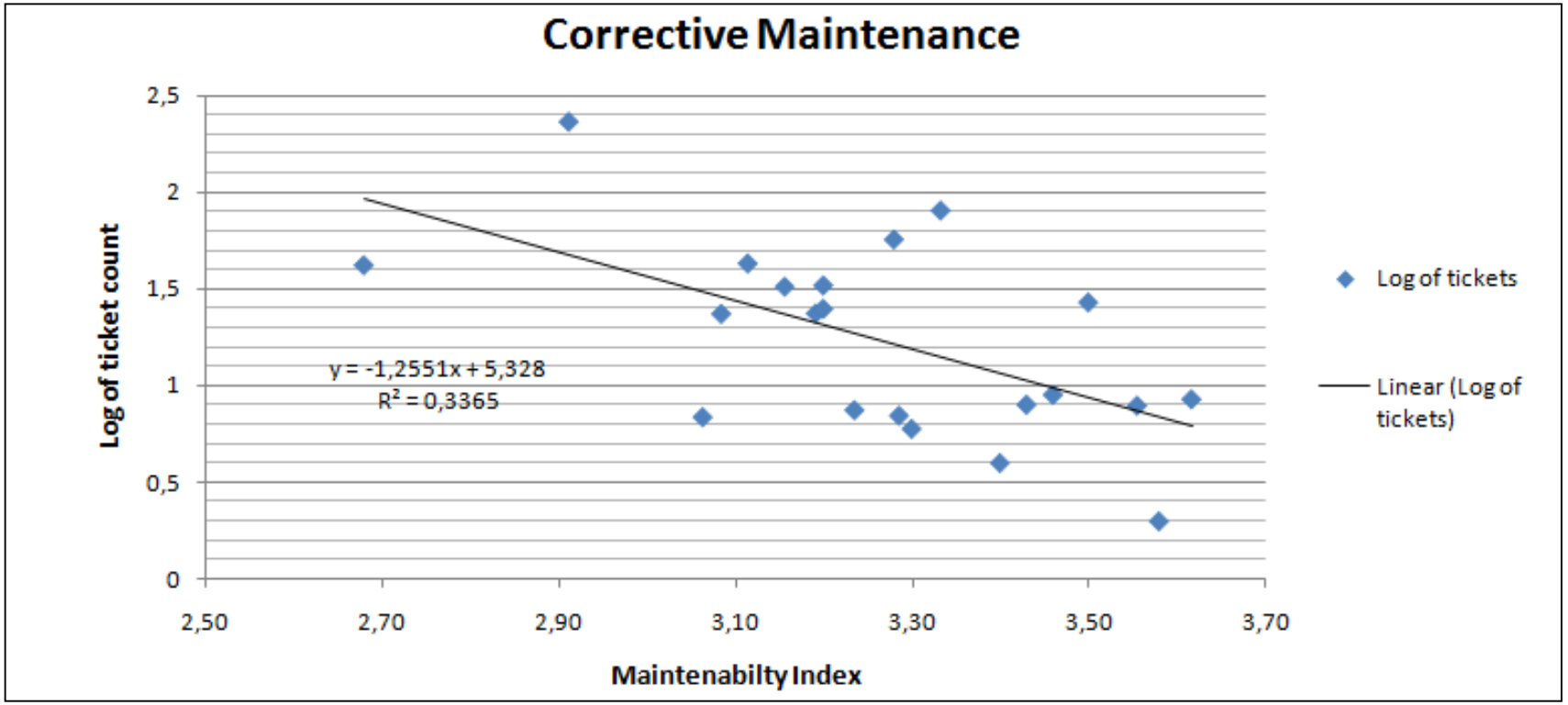
Measured impact in a complex enhancement-heavy environment

CLIENT STUDY OVER 24 MONTHS



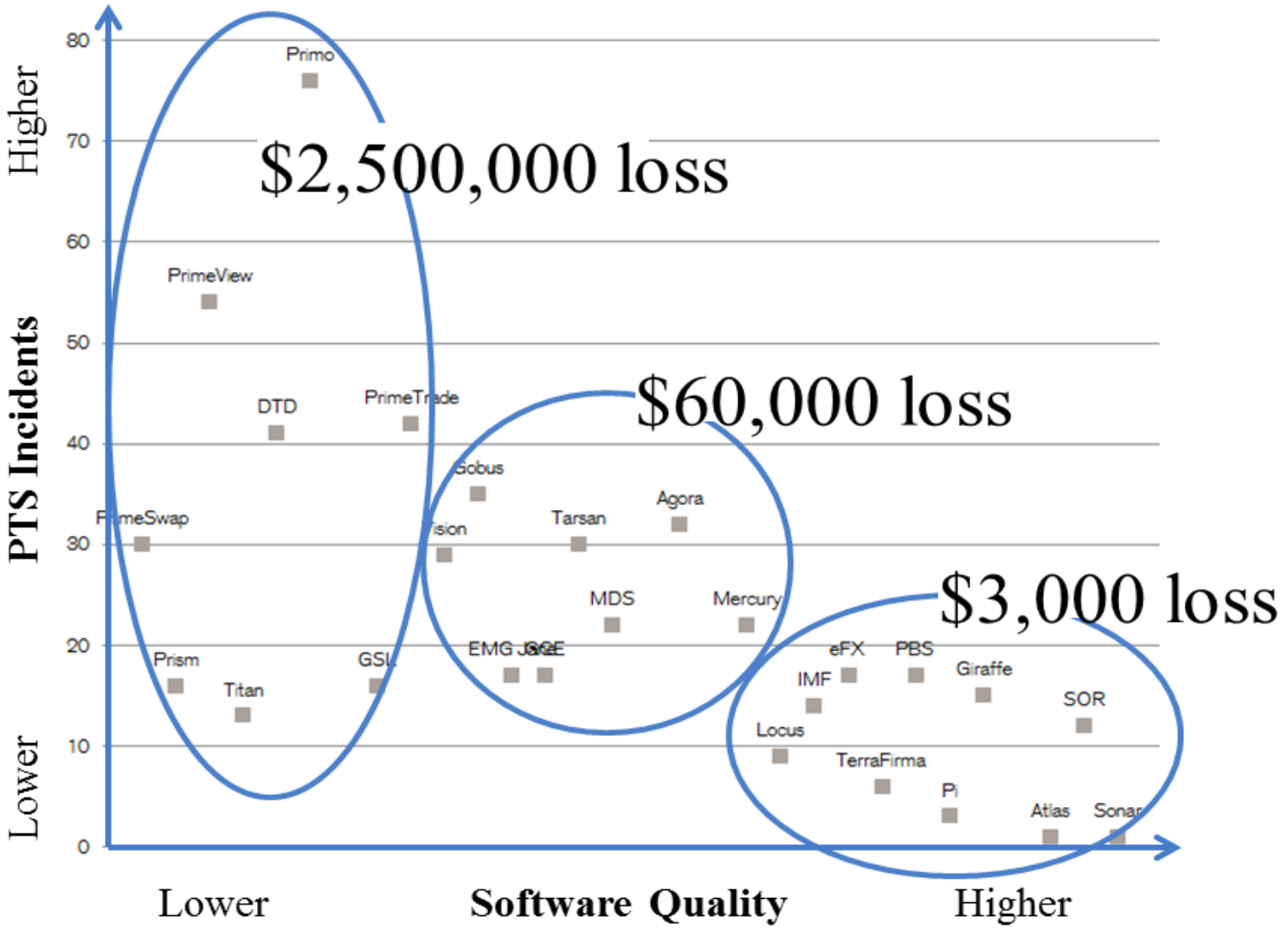
Tech Debt Reduction and Incident Rate

Correlation of maintenance effort with incident tickets across 20 customers of a global system integrator



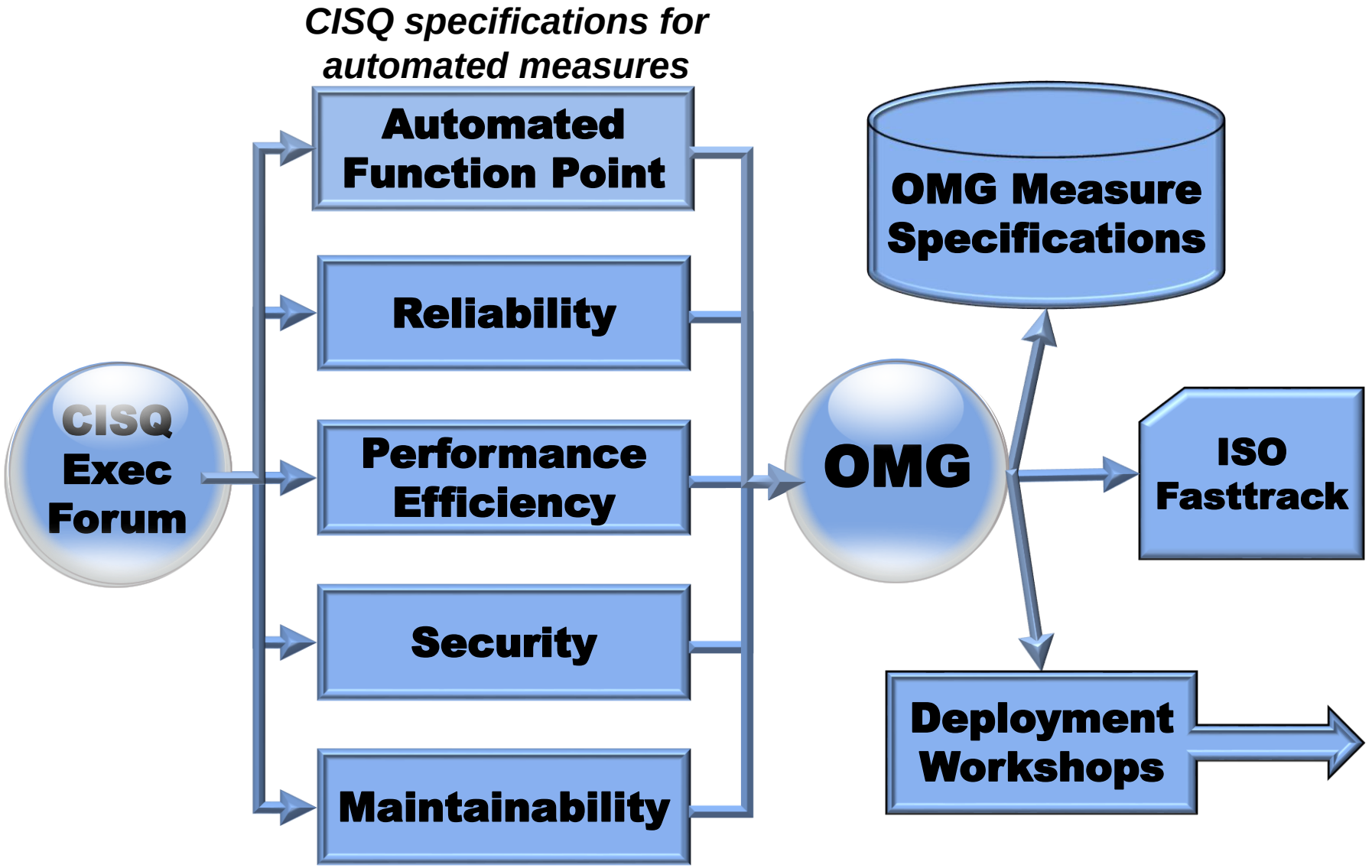
Increase of TQI by 0.24 = decrease in maintenance activity by 50%

Benefit of Tech Debt Reduction in Banking



CAST

Consortium for IT Software Quality (CISQ)



CISQ — Free Membership — www.it-cisq.org