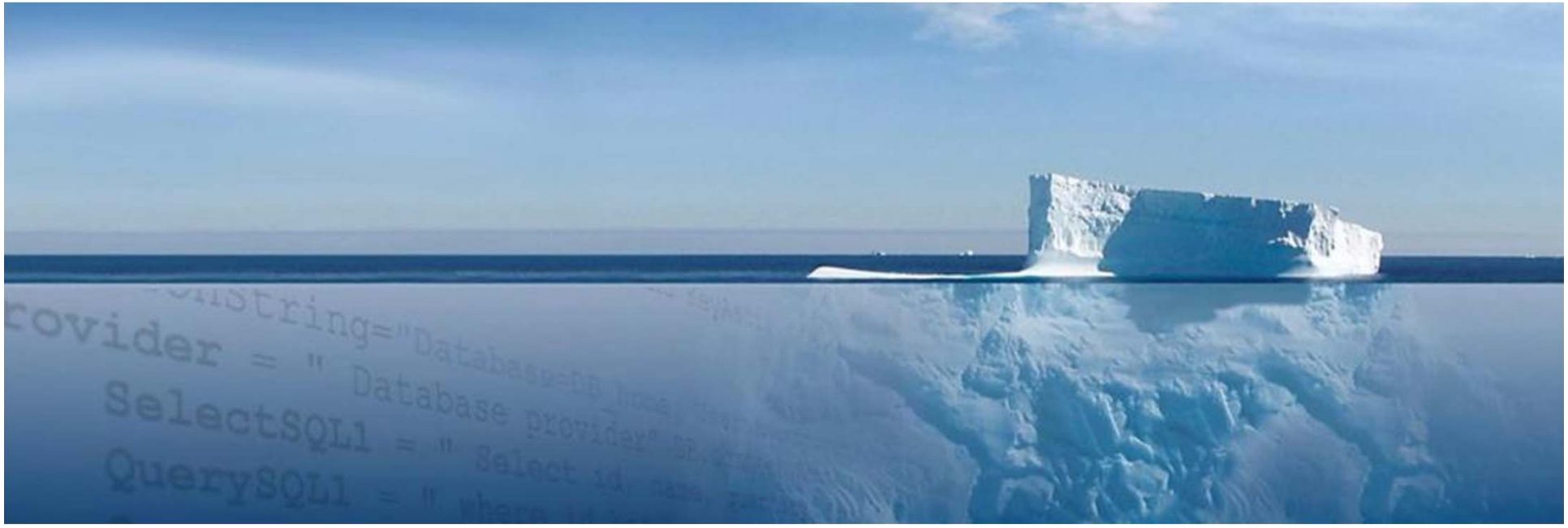
CAST

ACHIEVE INSIGHT. DELIVER EXCELLENCE.



CRASH: CAST Report on Application Software Health

Presented 16.11.15 at Software Technology Group, TU Dresden

Paul Nash, CAST GmbH, p.nash@castsoftware.com

CAST: Market Space Leader for Automated Software Analytics

250+ ENTERPRISE CUSTOMERS

TOP 10 SYSTEM INTEGRATORS

GLOBAL PRESENCE
USA, Germany, UK, France, Belgium, Italy, Spain, India

THE UNIT OF MEASURE FOR THOSE WHO BUY OR BUILD SOFTWARE

MARKET LEADER, PURE PLAYER, GLOBAL
NYSE Euronext

CAST products & services

ENTERPRISE SOFTWARE

CAST APPLICATION INTELLIGENCE PLATFORM

APPLICATION ANALYTICS DASHBOARD

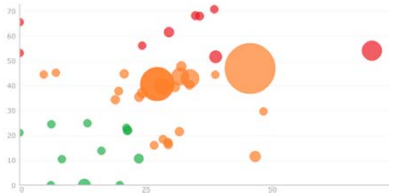
- OMG compliant software metrics
- Trend analysis
- Automated function points

ENGINEERING DASHBOARD

- Architectural analysis and blueprinting
- Critical violation drill down
- Transaction risk

SOFTWARE ANALYTICS SERVICES

CASTHIGHLIGHT



- SaaS, Cloud based.
- Rapid portfolio analysis
- Portfolio continuous monitoring

MANAGED SERVICES & CONSULTING

- Software analytics & engineering experts
- Software analytics as a service
- Value measurement & realization

INDUSTRY BENCHMARKS SERVICES

Application Software Quantified
appmarq

- Over 3,000 apps and 700 million LoC
- Query by industry, technology and geos
- CRASH Annual Report

CAST

Software Analytics - What do we measure?

RISK

MAINTAINABILITY

Risk of critical failures in production

Risk of performance / scaling issues

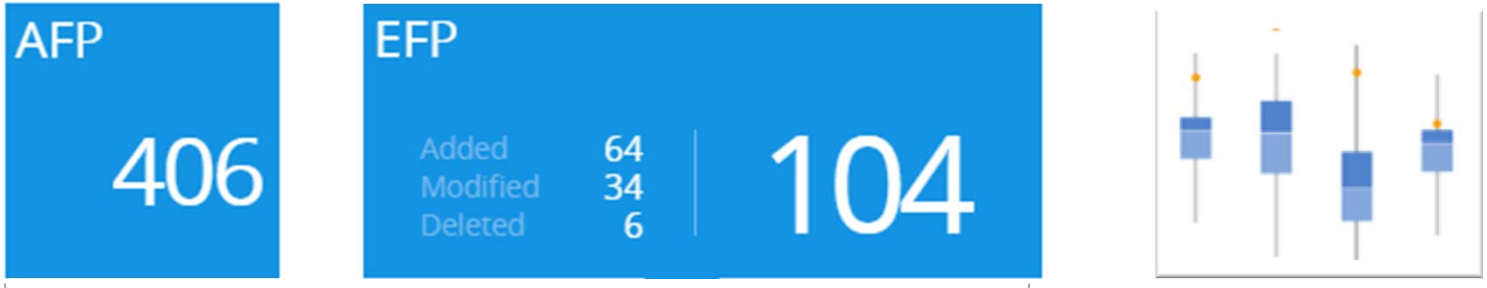
Risk of security breaches

Ease and speed of modifying

Ease and speed of learning

Overall application functional size

Performance and productivity

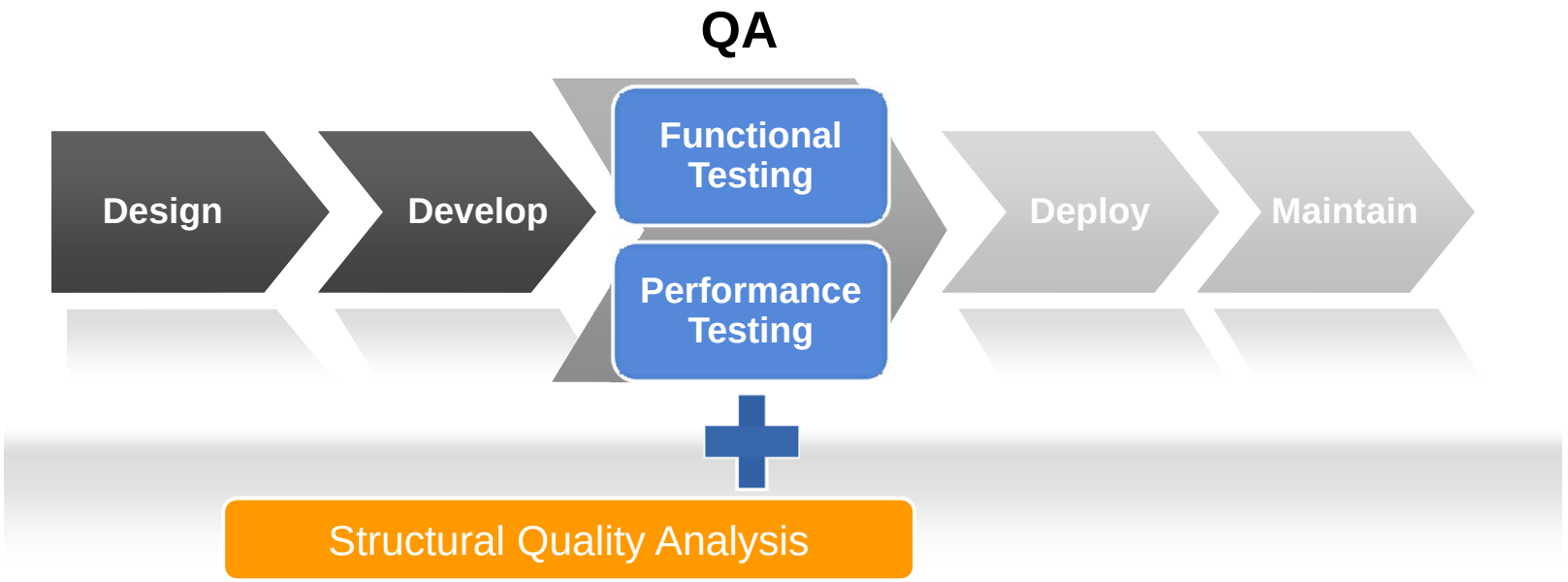


PRODUCTIVITY

BENCHMARK

CAST

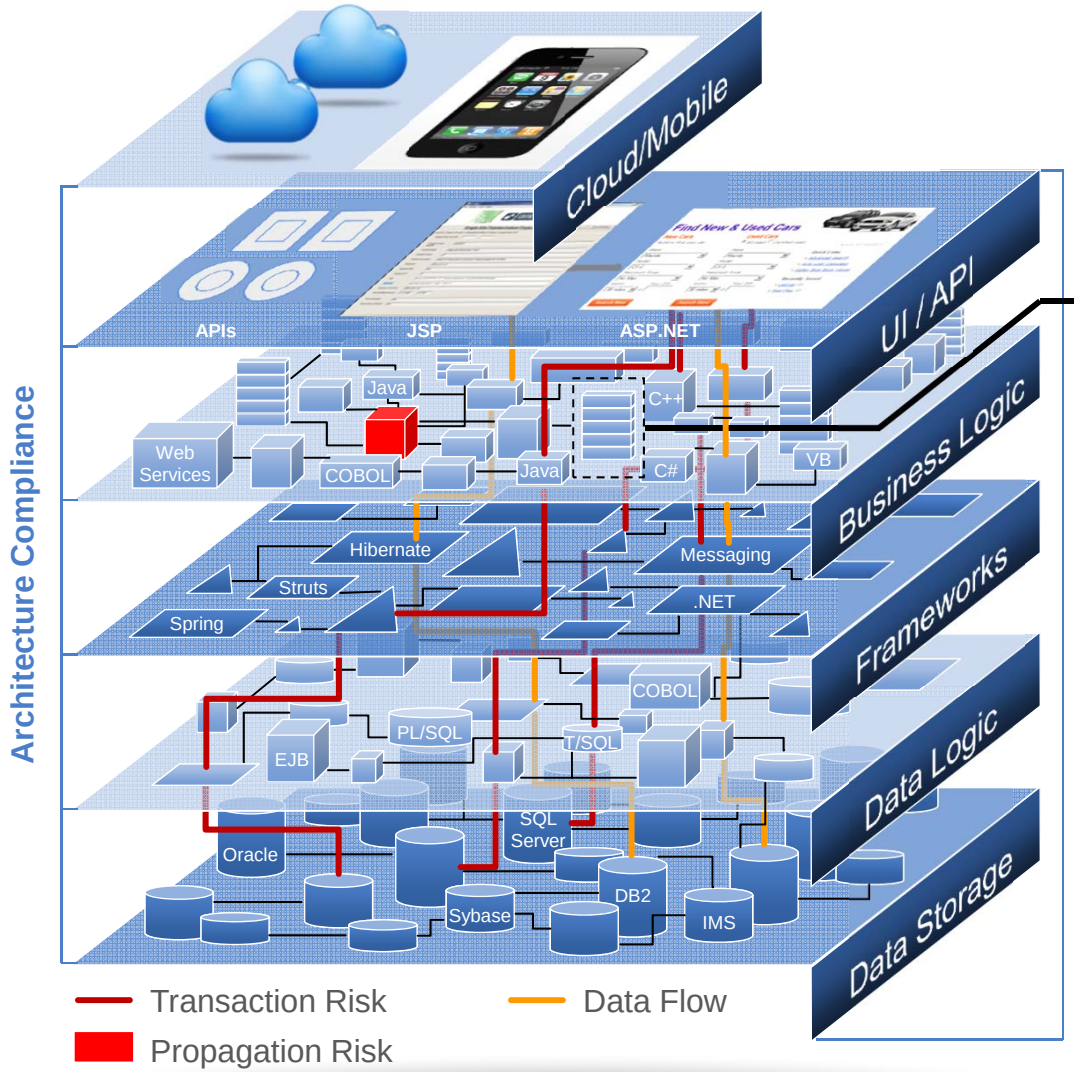
Evaluate Structural Quality of Software Deliveries



1. Verify delivery compliance with goals set in SLA
2. Identify sources of critical risk for immediate action
3. Define action plan for improvement in next deliveries

The Structural Quality evaluation is becoming a must-have for verifying and validating the code delivered from vendors

Software Analytics: How do we measure?



Program Unit Level

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

Technology Level

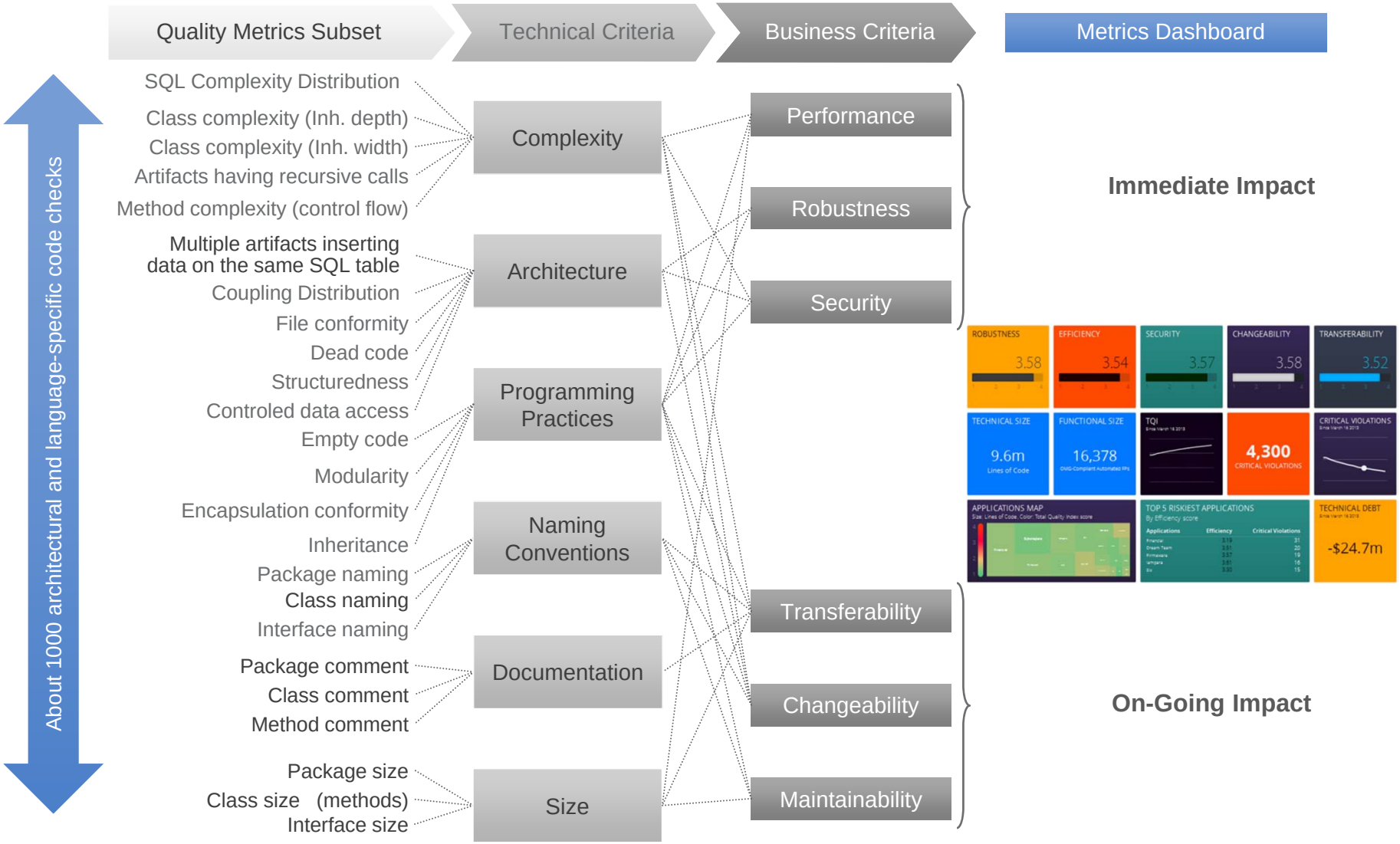
- Intra-technology architecture
- Intra-layer dependencies
- Intra-module communication
- Module complexity & cohesion
- Design & structure
- Inter-program invocation
- Security Vulnerabilities

System Level

- **Integration quality**
- **Architectural compliance**
- **Risk propagation simulation**
- **Application security**
- **Resiliency checks**
- **Transaction integrity**
- **Function point & EFP measurement**
- **Effort estimation**
- **Data access control**
- **SDK versioning**
- **Calibration across technologies**

CAST

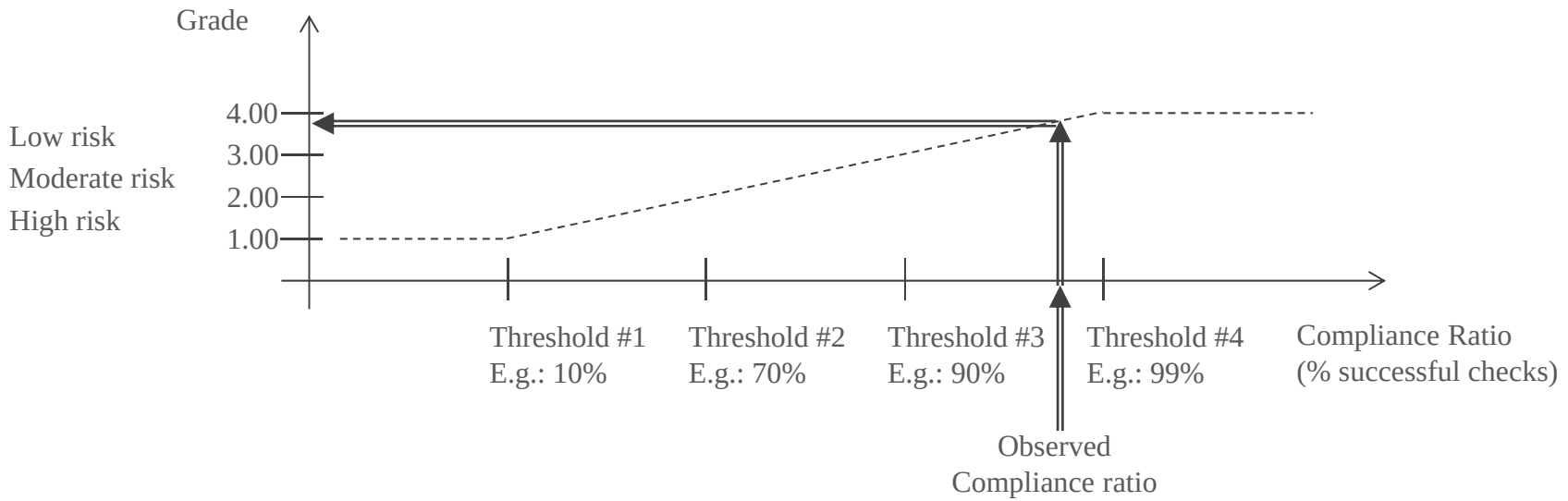
Overview of CAST Quality Model



Computation of Quality Scores

$$\text{Compliance Ratio} = \frac{\text{successful checks}}{\text{successful checks + failed checks}} \times 100$$

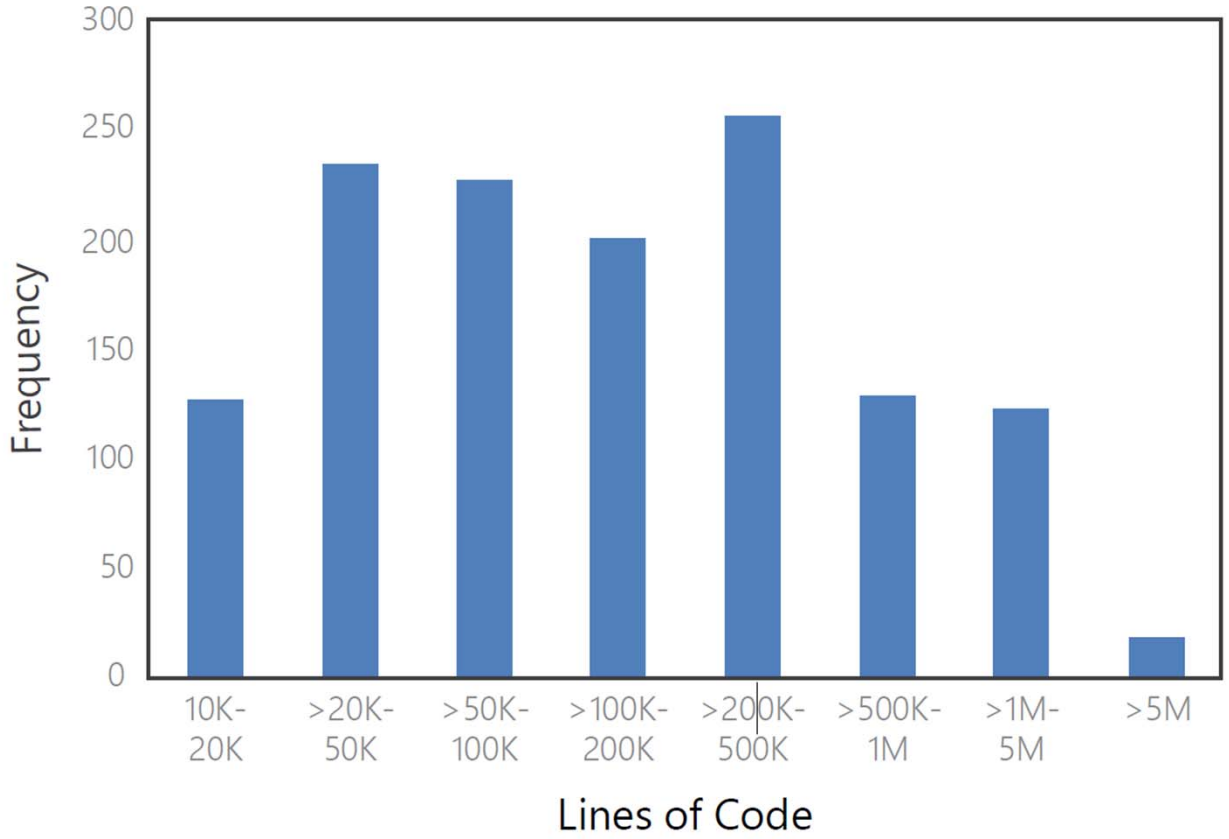
(% of Successful checks)



CAST

CRASH Data Sample: Size Distribution

Figure 1. Distribution of applications by size categories



- 1316 applications, mostly business critical
- 212 organizations worldwide, from 12 different verticals
- 706 MLOC total code volume
- 20 large enterprise systems over 5 MLOC

CAST

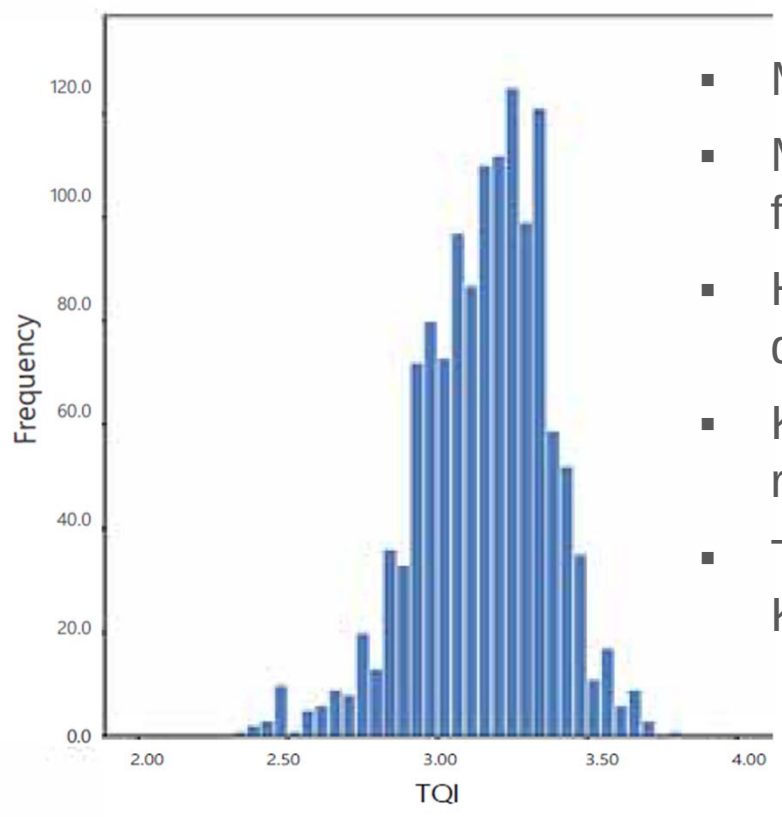
CRASH Data Sample: Distribution Across Languages

Table 1. Descriptive statistics for application size within language categories

Statistic	J-EE	Cobol	.NET	ABAP	Oracle Froms	Oracle ERP	C	C++	ASP	Mixed	Sample
Sample	565	280	127	77	59	33	39	28	24	84	1,316
Mean	363	681	303	669	292	333	541	340	149	844	471
Std. Dev	914	1,446	385	798	517	894	1,797	595	309	1,664	1,093
Maximum	10,098	10,980	2,388	2,986	2,338	4,867	11,302	2,870	1,471	9,639	11,302
75th % - ile	323	482	408	901	279	144	336	357	82	719	390
Median	106	153	181	330	82	54	116	115	40	211	128
25th % - ile	35	63	57	72	34	35	48	37	31	83	43
Minimum	10	10	12	15	10	16	13	12	15	11	10

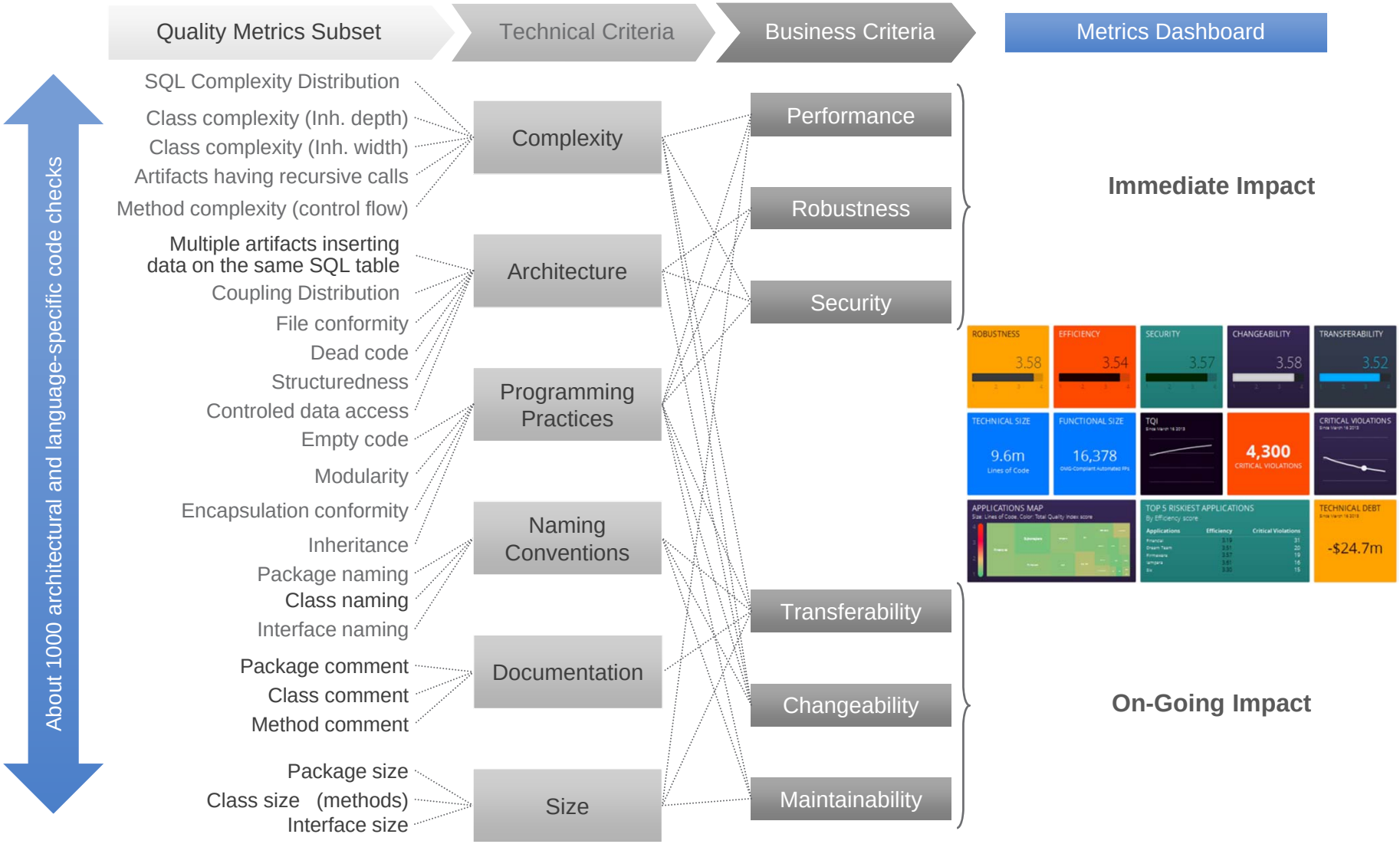
- Largest number of apps in Java-EE, COBOL, .NET, Oracle, and ABAP
- Median size in each technology is half or less the size of the mean due to large apps accounting for most of the total LOC
- "Mixed" is usually a mix of COBOL and Java-EE, usually in Financial and Telco verticals (customer-facing web apps on top of legacy applications)

CRASH Data Sample: Global Quality Characteristics



- Most of the scores are above 3
- May be due to the high dependence on quality for large critical apps
- However, there's a "tail" of apps with scores down to around 2
- Keep in mind that apps are in production, this requires minimum quality at least
- TQI distribution is similar to distribution of other KPIs (Performance, Robustness etc.)

Overview of CAST Quality Model



CRASH Data Sample: Shared Variance in KPIs

Table 4. Percent of shared variance among health factors, Total Quality Index, and size

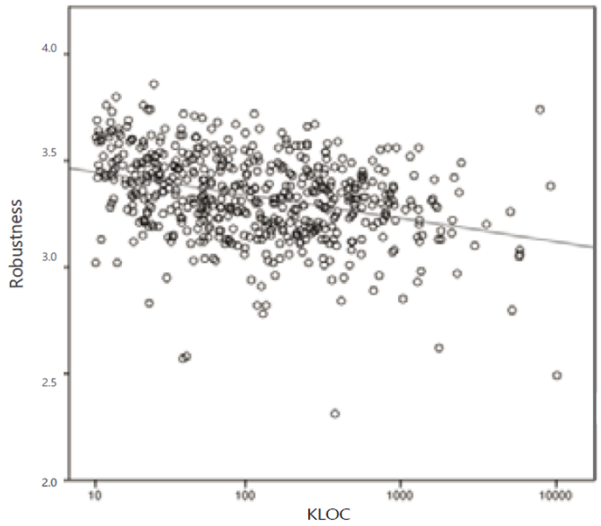
	TQI	Robust	Perform	Security	Change	Transfer	KLOC
Robustness	72		10	36	38	34	5
Performance	32			5	14	13	0
Security	37				2	7	2
Changeability	56					30	1
Transferability	61						1

Correlations underlying all r^2 s $\neq 0$ are significant at $p < .001$, n= 1298

- Percent of shared variance is the square of the correlation coefficient between two variables and measures the strength of their relationship
- Security and Robustness are strongly related, so good architectural and coding practice seem to affect both in parallel
- Performance has weak relationships with the other KPIs, so the believe that improving performance affects the other KPIs negatively appears to be a myth

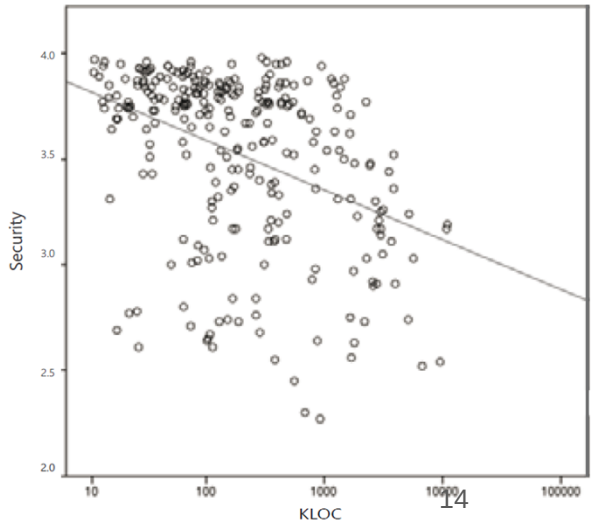
CRASH Data Sample: KPI Variation with App Size

Figure 3. Scatterplot of Robustness scores with size in Java-EE



- For Java-EE, Robustness declines slightly with size
- This is surprising since the JEE architecture is built to handle scaling effects, so wrong usage?

Figure 4. Scatterplot of Security scores with size in COBOL



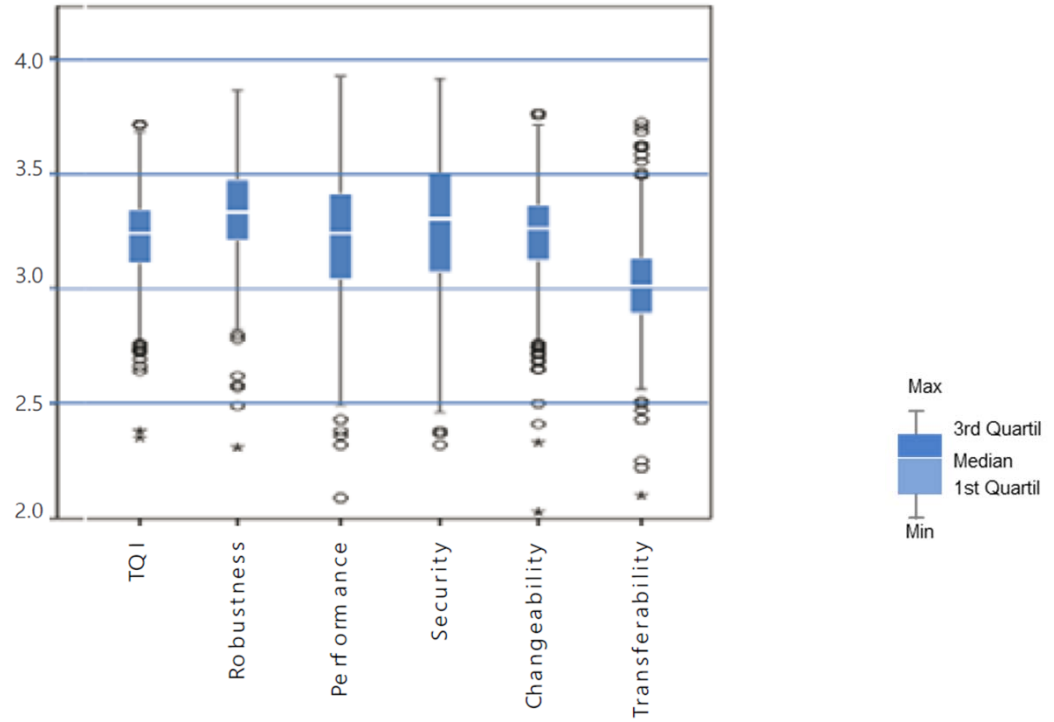
- For COBOL, Security drops slightly with application size
- This is from a high absolute value, see below

- All other KPIs have little variation with application size

CAST

CRASH Results by Language: Java-EE

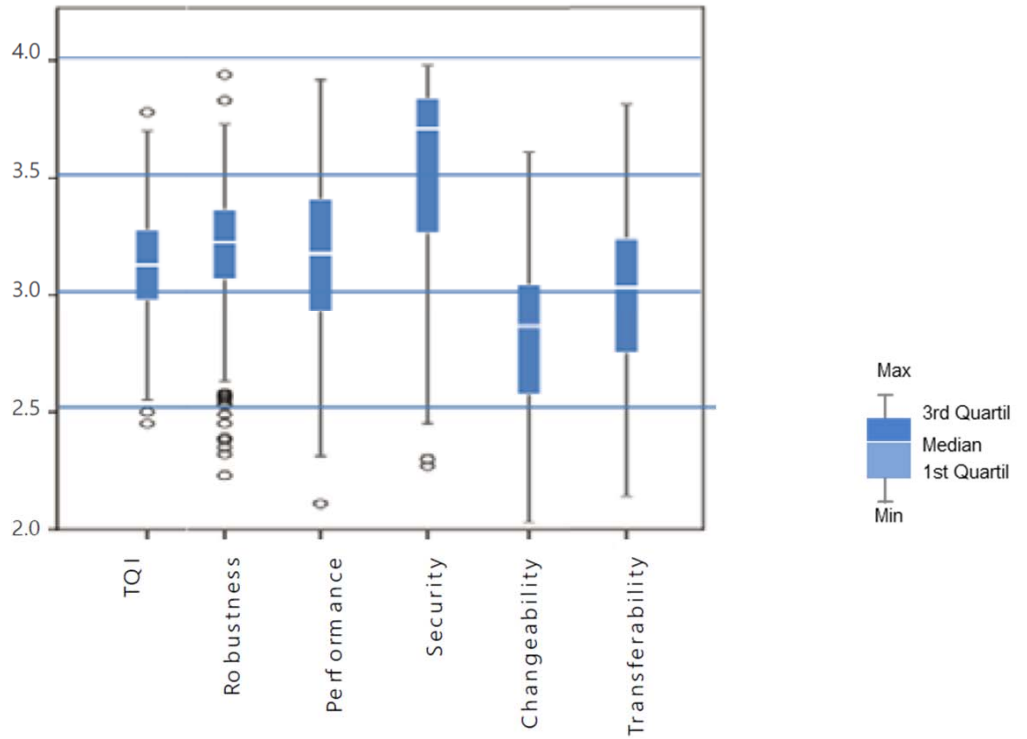
Figure 5. Box and whisker plots for TQI and health factors in Java-EE



- Transferability significantly lower than other KPIs (complex constructs, no doc?)
- Lots of variation in all KPIs, many outliers
- This is concerning, especially for Security with Internet applications

CRASH Results by Language: COBOL

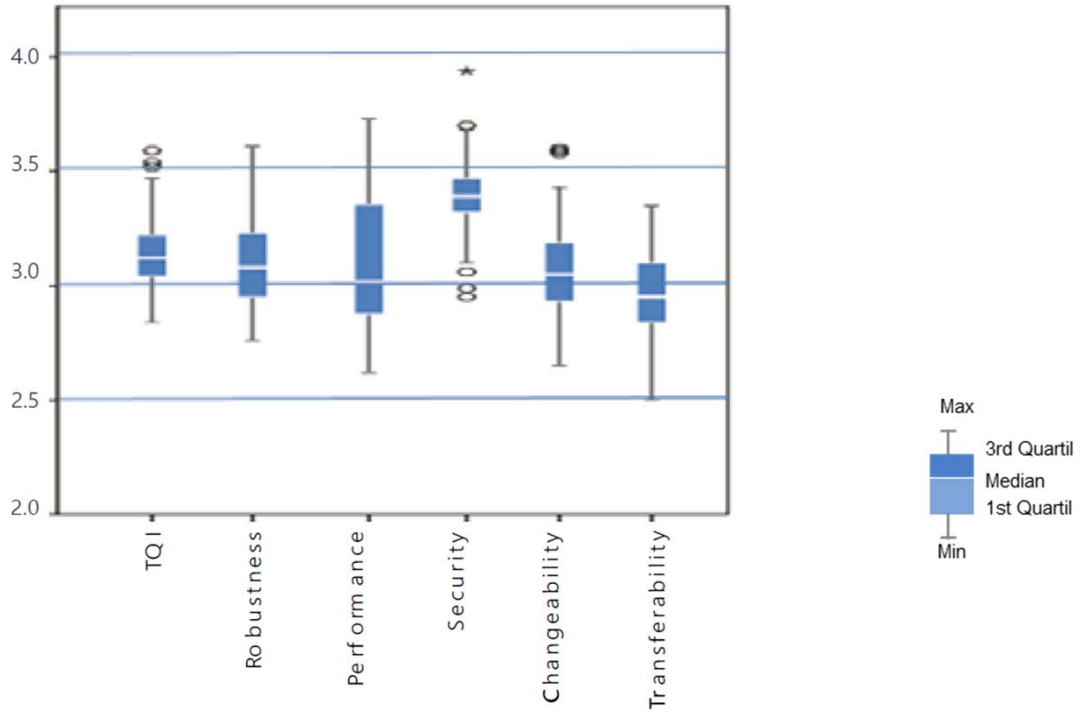
Figure 6. Box and whisker plots for TQI and health factors in COBOL



- Security significantly higher than other KPIs (due to finance industry needs?)
- Changeability/Transferability lower, probably due to module size and complexity
- Average module size in CRASH sample for COBOL is 600 LOC, while in most modern languages it is 50 LOC (30 LOC average for Java-EE)

CRASH Results by Language: APAP (SAP)

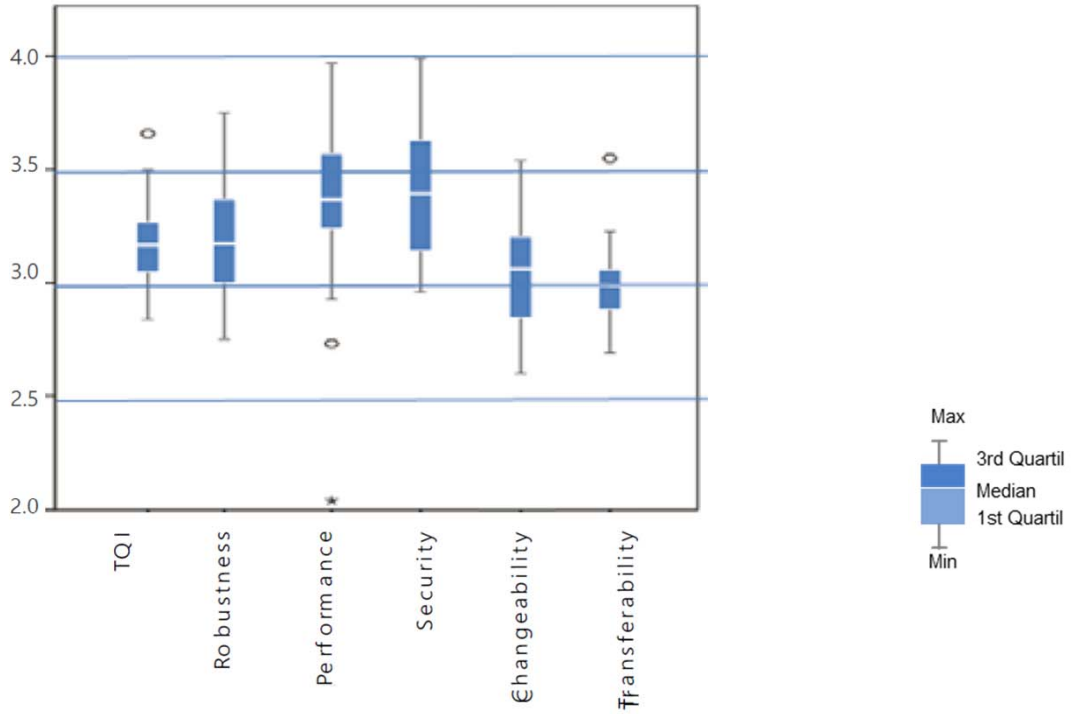
Figure 8. Box and whisker plots for TQI and health factors in ABAP



- ABAP is a language used for customizing applications built atop the SAP application platform
- Security higher than other KPIs (finance requirement? platform restrictions?)
- Largest variation for Performance, a known issue in the SAP user base

CRASH Results by Language: C++

Figure 12. Box and whisker plots for TQI and health factors in C++



- Security and Performance substantially higher than other KPIs
- C++ often chosen for performance-sensitive applications
- Changeability and Transferability were the lowest across languages
- Likely due to complex constructs with machine-accessible attributes using C++

CAST

CRASH Sample Variations by different Demographics

Figure 14. Health factor distributions by industry sector

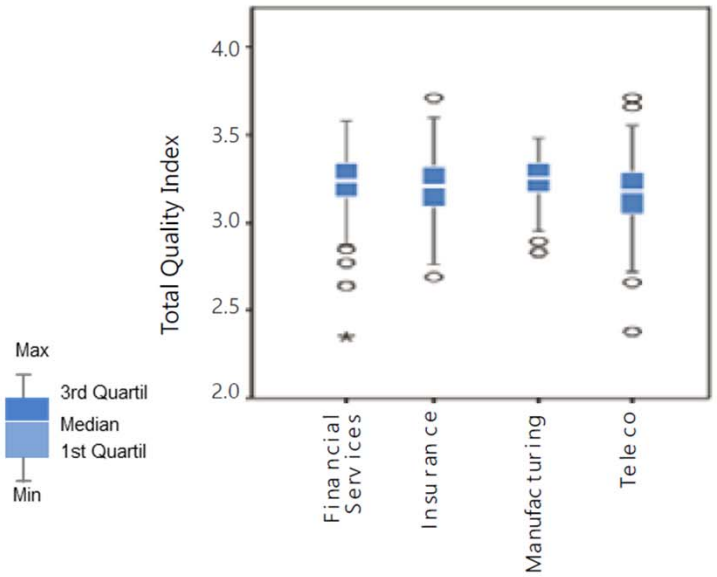


Figure 16. Health factor distributions for onshore and offshore applications

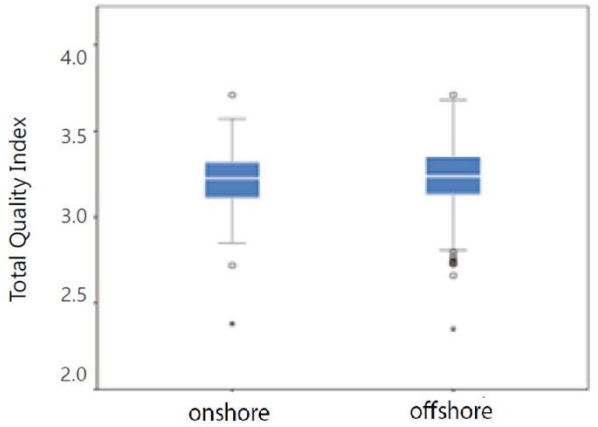
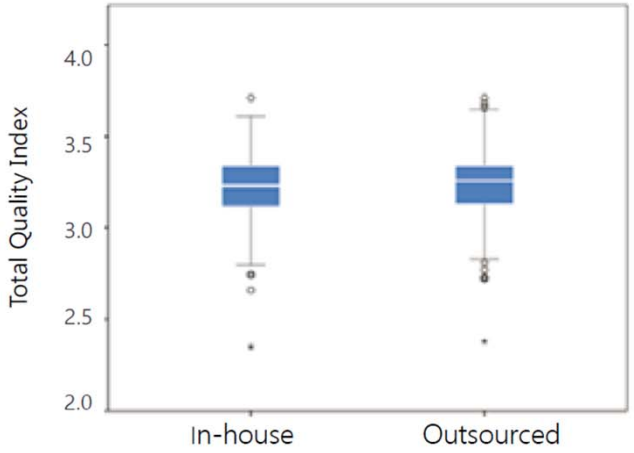


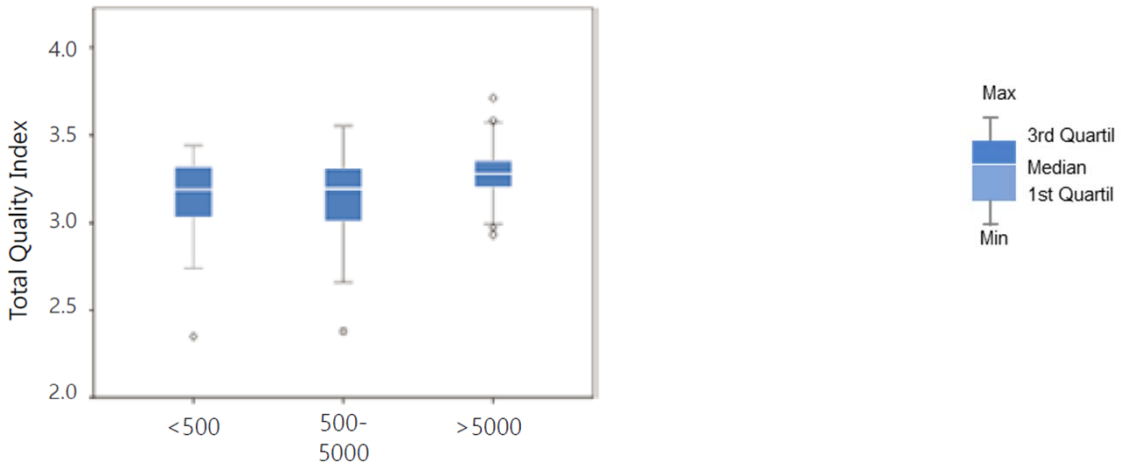
Figure 15. Health factor distributions for in-house and outsourced applications



- Very little variation of KPIs by Vertical, In- vs. Outsouced, and On- vs. Off-Shore
- To avoid influence by language, only Java-EE was used
- Changeability and Robustness slightly better for On-Shore

CRASH Sample Variations by Number of Users

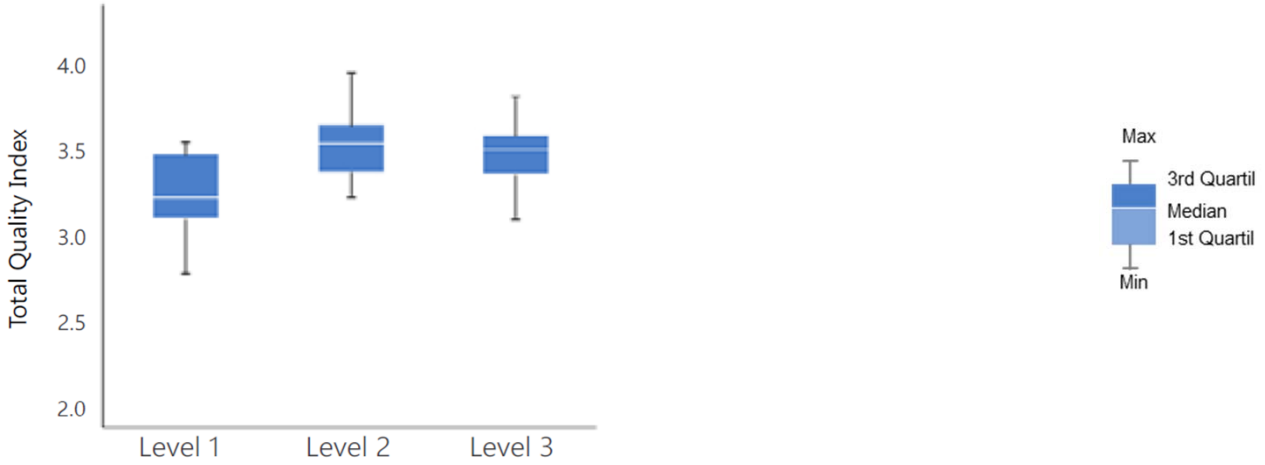
Figure 19. Health factor distributions for number of end users



- For all the health factors, the significant differences were accounted for by the higher scores for applications serving more than 5000 users
- Applications serving more than 5000 users are typically customer facing
- Not surprisingly, greater effort would be focused on the structural quality of these applications considering their risk to the business if they suffer operational problems or are difficult to maintain

CRASH Sample Variations by CMMI Level

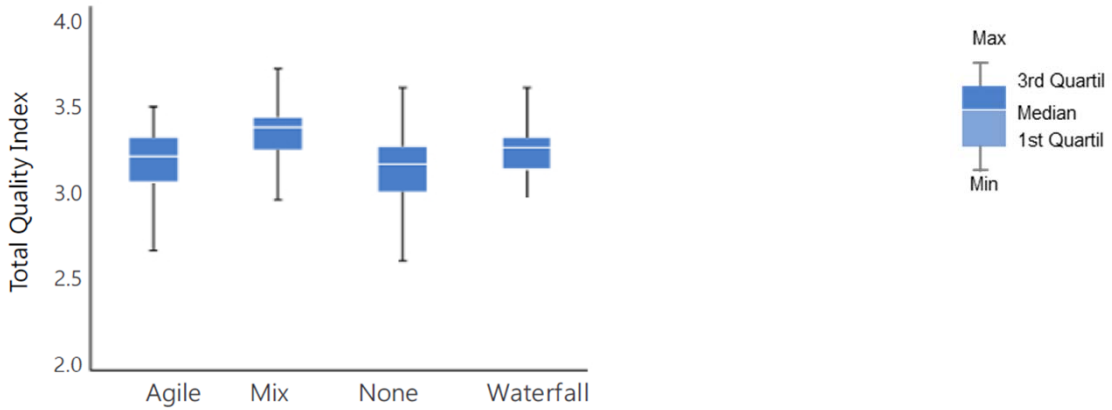
Figure 17. Health factor distributions for CMMI Levels 1, 2, and 3 applications



- The CMMI (Capability Maturity Model) relates to the degree of formality and optimization of processes: 1 = Initial (chaotic, ad hoc, individual heroics), 2 = Repeatable (sufficiently documented process for repeatability), 3 = Defined, 4 = Managed, 5 = Optimizing
- Strong dependency of all KPIs on CMMI level, expecially from 1 to 2
- This reflects the common knowledge that removing obstacles like unachievable commitments and volatile requirements lets developers perform their work in a more orderly and professional manner

CRASH Sample Variations by Development Method

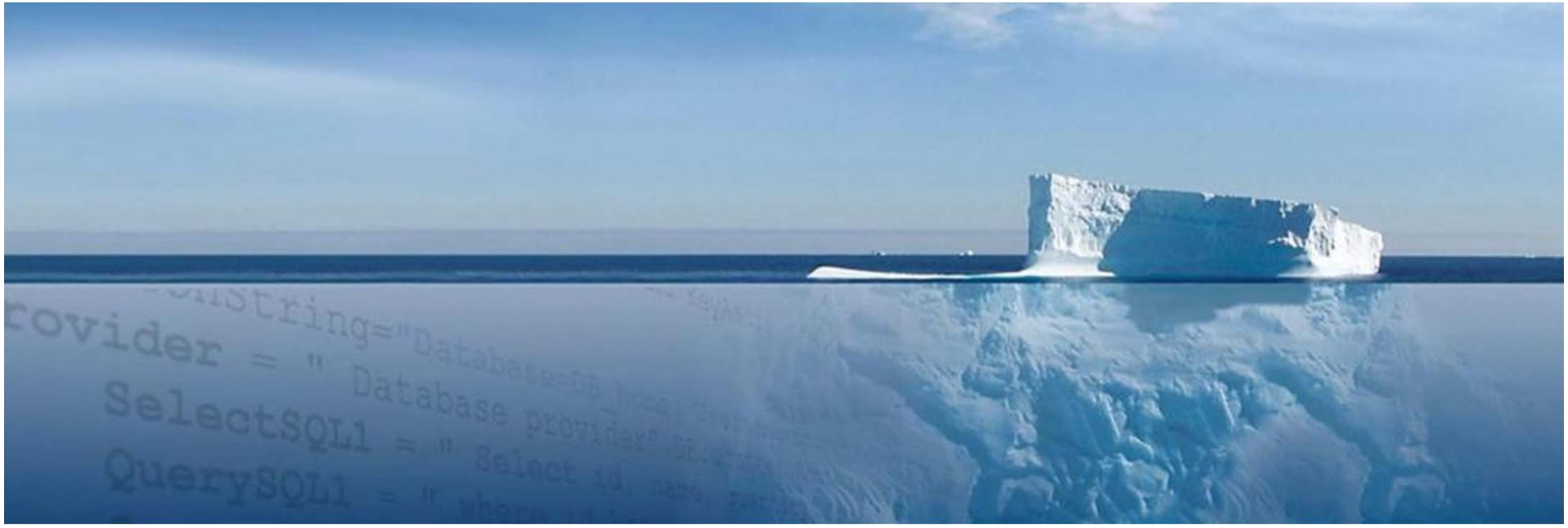
Figure 18. Health factor distributions for development methods



- No usage of a development model yields weakest results
- Agile and waterfall methods are almost at the same level
- A mix of Agile and waterfall methods creates significantly better quality
- This reflects the experience that while the rapid development of new functionality profits from agile environments, overall technical requirements like scalability require a more stable, predefined architecture that adresses these requirements by design (platform concept)

CAST

ACHIEVE INSIGHT. DELIVER EXCELLENCE.



CRASH: CAST Report on Application Software Health

Presented 16.11.15 at Software Technology Group, TU Dresden

Paul Nash, CAST GmbH, p.nash@castsoftware.com