The CRASH Report 2014-2015
(CAST Research on Application Software Health)

The Global State of Structural Quality in IT Applications

CAST
Achieve Insight. Deliver Excellence.

Executive Summary

This is the third report produced by CAST on global trends in the structural quality of business application software. Structural quality refers to the engineering soundness of the architecture and coding of an application, rather than to the correctness with which it implements the customer's functional requirements. These reports highlight trends in five structural quality characteristics—Robustness, Performance, Security, Changeability, and Transferability. The data in this report are drawn from the Appmarq benchmarking repository maintained by CAST. The sample in this report consists of 1316 applications submitted by 212 organizations in 12 industry sectors primarily in the United States, Europe, and India. These applications totaled approximately 706 million lines of code.

The distribution of scores for the operational risk factors of Robustness, Performance, and Security are higher than for the cost-related factors of Changeability and Transferability. Correlational analysis indicated that the violations of good architectural and coding practice that reduce an application's robustness are related to the types of violations that make it less secure. In both COBOL and ABAP the scores for Security were higher than those for the other health factors, perhaps reflecting the greater tendency for applications written in these languages to manage financial information.

Statistical analysis found that:

- With minor exceptions, health factor scores have little relation to application size.
- CMMI Level 1 organizations produced applications with substantially lower structural quality on all health factors than applications developed in CMMI Level 2 or Level 3 organizations.
- Across all health factors, a hybrid mix of Agile and Waterfall methods produced higher scores than either Agile or Waterfall methods alone.
- The choice to develop applications in house versus outsourced had no effect on health factor scores, while the choice to develop applications onshore versus offshore had very small effects on Changeability and Robustness.

Structural quality on large business critical applications was best achieved when impediments to disciplined software engineering practices were removed and early design activity was integrated with short cycle releases. Process maturity and development method were more important than where or by which organization the application was developed.

Contents

1. Introduction

1.1. Overview

This is the third annual report produced by CAST on global trends in the structural quality of business application software. Structural quality refers to the engineering soundness of the architecture and coding of an application, rather than to the correctness with which it implements the customer's functional requirements. These reports highlight trends in five structural quality characteristics—Robustness, Security, Performance, Transferability, and Changeability. Structural quality is measured as violations of rules representing good architectural and coding practice in each of these five areas. Structural analysis technology can evaluate an application for violations of structural quality rules that are difficult to detect through standard testing. Structural quality flaws are the defects most likely to cause operational problems such as outages, performance degradation, unauthorized access, or data corruption. This report provides an objective, empirical foundation for discussing the structural quality of software applications throughout industry and government.

1.2. Sample

The data in this report are drawn from the Appmarq benchmarking repository maintained by CAST, comprised of 1316 applications submitted by 212 organizations for the analysis and measurement of their structural quality characteristics. These applications totaled approximately 706 MLOC (million lines of code). These organizations are located primarily in the United States, Europe, and India. The 2014 sample includes almost double the 2012 sample of 745 applications and 365 MLOC (million lines of code), and was submitted by one-third more organizations.

The sample is widely distributed across size categories and appears representative of the types of applications in business use. However, the applications usually submitted for structural analysis and measurement tend to be the most business critical systems, so we do not claim that this sample is statistically representative of all the world's business applications.

Figure 1 displays the distribution of applications over eight size categories measured in lines of code. The applications range from 10 KLOC (kilo or thousand lines of code) to just over 11 MLOC. Within this distribution 28% of the applications are less than 50 KLOC, 33% contain between 50 KLOC and 200 KLOC, 29% contain between 201 KLOC and 1 MLOC, and 11% are over 1 MLOC including 20 applications over 5 MLOC.

Table 1 reports the descriptive statistics for size in thousands of lines of code within each size category and for the entire sample. The largest number of applications in this sample were developed in Java-EE, COBOL, .NET, Oracle, and ABAP. For each technology, Table 1 presents the mean and median size of applications as well as the size of the largest application. Note that the median size of applications in each technology is half or less the size of the mean, confirming the strong positive skews in application sizes.

Figure 1. Distribution of applications by size categories

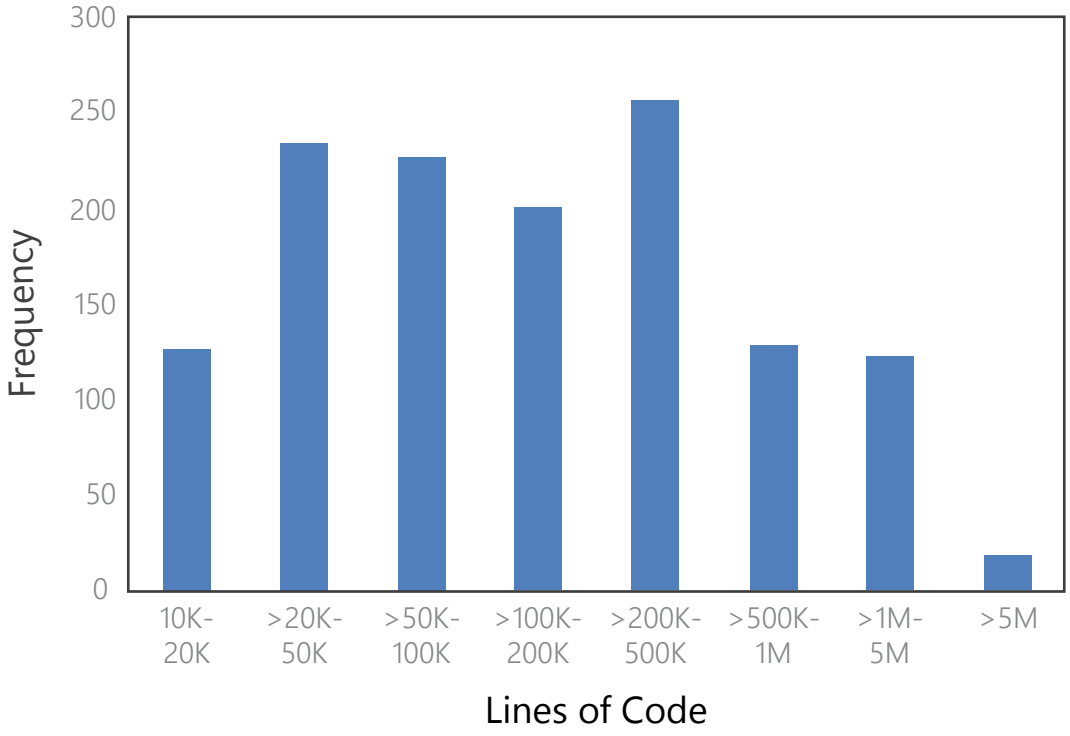


Table 1. Descriptive statistics for application size within language categories

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

These applications are large with the median size typically being over 100 KLOC in each technology except ASP and Oracle. At least one application in every language contained over 1 MLOC, while the largest applications in Java, COBOL, and C contained over 10 MLOC. On average, the largest applications were developed in COBOL, ABAP, and a category labeled 'Mixed.'

The 'Mixed' category contains applications that have large subsystems written in different languages. Most often this category includes a mix of COBOL and Java-EE, but other languages or technologies included in 'Mixed' are DB2, Delphi, Pacbase, Peoplesoft, PHP, PL1, Powerbuilder, Powercenter, RDL, RPG, script languages, Siebel, SQL Server, and Visual Basic. As is evident in Table 2, applications in our sample with mixed technologies were primarily found in financial services and telecommunications. Even though there are 84 such applications they are not analyzed further in this report because of the difficulty in interpreting mixed language results. However, since many of these applications were a mix of COBOL and Java-EE in financial services, the results from our preliminary analyses of these applications were frequently similar to results from COBOL.

There are 12 industry sectors represented in the 212 organizations that submitted applications to the Appmarq repository. Table 2 displays a breakdown of the applications distributed by language across the industry sectors. Some trends observable in these data include the heaviest concentration of ABAP applications in manufacturing, while CO-

Table 2. Frequency of applications by language category within each industry sector

Industry	Orgs.	Apps.	J-EE	Cobol	.NET	Oracle Froms	Oracle ERP	Mixed	ABAP	C	C++	ASP	KLOG
Financial Serv.	51	421	179	146	20	17	5	40			6	8	299,249
Insurance	34	314	61	97	26	5	9	4	1	4	1	6	113,930
Telecom	19	187	106	1	22	9	2	16	7	14	6	4	62,786
Manufacturing	25	169	65	14	17	4	8	5	49	6	1		55,787
Utilities	15	56	34		1	3	2	3	1	8	4		65,787
Government	17	56	36	1	1	1	1	3	8	1		4	25,356
Retail	12	48	18	3	9	5	1	4	6	2			31,076
IT Consulting	17	41	15	4	9	8	0	2	1		2		25,570
Business Serv.	2	40	13	13	6					1	5	2	10,740
Software ISV	12	32	17		5	4		5			1		35,971
Energy	12	22	6	1	6	1	1	1	4		2		11,671
Other	12	30	15		5	2	4	1		3			5,742
Total	212	1316	565	280	127	59	33	84	77	39	28	24	705,935

BOL applications were concentrated most heavily in financial services and insurance. Java-EE applications accounted for at least one-third of the applications in every industry sector except insurance.

1.3. Structural Quality Terminology and Measures

The following terms will be used through this report.

Structural Quality: The non-functional quality of a software application that indicates how well the code is written from an engineering perspective. It is sometimes referred to as technical quality or internal quality, and represents the extent to which the application is free from violations of good architectural or coding practice.

Structural Quality Health Factors: Data reported here involve the five structural quality characteristics defined below. Scores for these measures are computed on a scale of 1 (high risk) to 4 (low risk) by analyzing the application to detect violations of over 1200 rules of good architectural and coding practice. Scoring is based on an algorithm that evaluates the number of times a violation occurred compared to the number of opportunities where it could have occurred, weighted by the severity of the violation and its relevance to each individual quality characteristic. The quality characteristics reported here include:

- Robustness: The stability and resiliency of an application and the likelihood of introducing defects when modifying it.
- Performance: The efficiency of the software with respect to processing time and resources used.
- Security: An application's ability to prevent unauthorized intrusions.
- Changeability: An application's ability to be easily and quickly modified.
- Transferability: The ease with which a new team can understand an application and become productive working on it.
- Total Quality Index: A composite score computed from the five quality characteristics listed above.

Violations: A structure or anti-pattern in the source code that is inconsistent with good architectural or coding practice and has proven in the past to cause problems that affect either the cost or risk of an application.

2. Quality Characteristic Results for the Full CRASH Sample

2.1. Health Factor Distributions

The distribution of scores for all Health Factors and the Total Quality Index are presented in Figure 2 and the descriptive statistics drawn from these distributions are presented in Table 3. The scores for the health factors are not directly comparable since scores for each health factor are computed from different numbers of violations. However, comparing distributional shapes is revealing.

First, the distributions for Robustness, Security, and Changeability are negatively skewed indicating the preponderance of scores being in the upper range. For instance, although the mean and median scores for Security are relatively high, there are numerous applications in the lower tail of the distribution whose significantly lower scores should be of concern to Information Security Officers. Based on our field experience, scores below 3.0 on Reliability, Security, or Performance Efficiency should be addressed.

Approximately 75% of scores for the operational risk factors of Robustness, Performance, and Security are above 3.0, compared to the lower distributions for the cost-related health factors of Changeability and Transferability. Among possible explanations are that fewer violations related to operational risk are released from development, or these violations are prioritized for remediation over the cost-related factors of Transferability and Changeability. Since the Total Quality Index is a composite of the five health factor scores, its distribution and descriptive statistics tend toward a mean among the statistics for each of its health factor constituents, with the exception that its range and standard deviation are smaller. Thus, this composite score exhibits less variation and is less affected by outliers or extreme scores.

The relationships among the five health factors, the Total Quality Index, and size are presented in Table 4 as the percent of shared variance between each pair of variables. The percent of shared variance is the square of the correlation coefficient between two variables, and thus measures the strength of their relationship. By directly assessing the extent to which variation in one variable is related to variation in the other variable, the percent of shared variance allows the magnitudes of relationships to be more easily compared than they are by the correlation coefficient. Four observations emerge from these relationships.

The first observation is that while Security shares over a third of its variance with Robustness, its relationships with the other health factors are weak, typically sharing less than 7% of their variance. Thus, the violations of good architectural and coding practice that reduce an application's Robustness are related to the types of violations that make it less secure. However, the relationship of Security weaknesses with the types of violations that affect an application's Performance, Changeability, or Transferability is weak.

The second observation is that Performance has weak relationships with the other health factors, sharing only between 5% and 14%

Figure 2. Distributions of Health Factor scores for the full 2014 sample

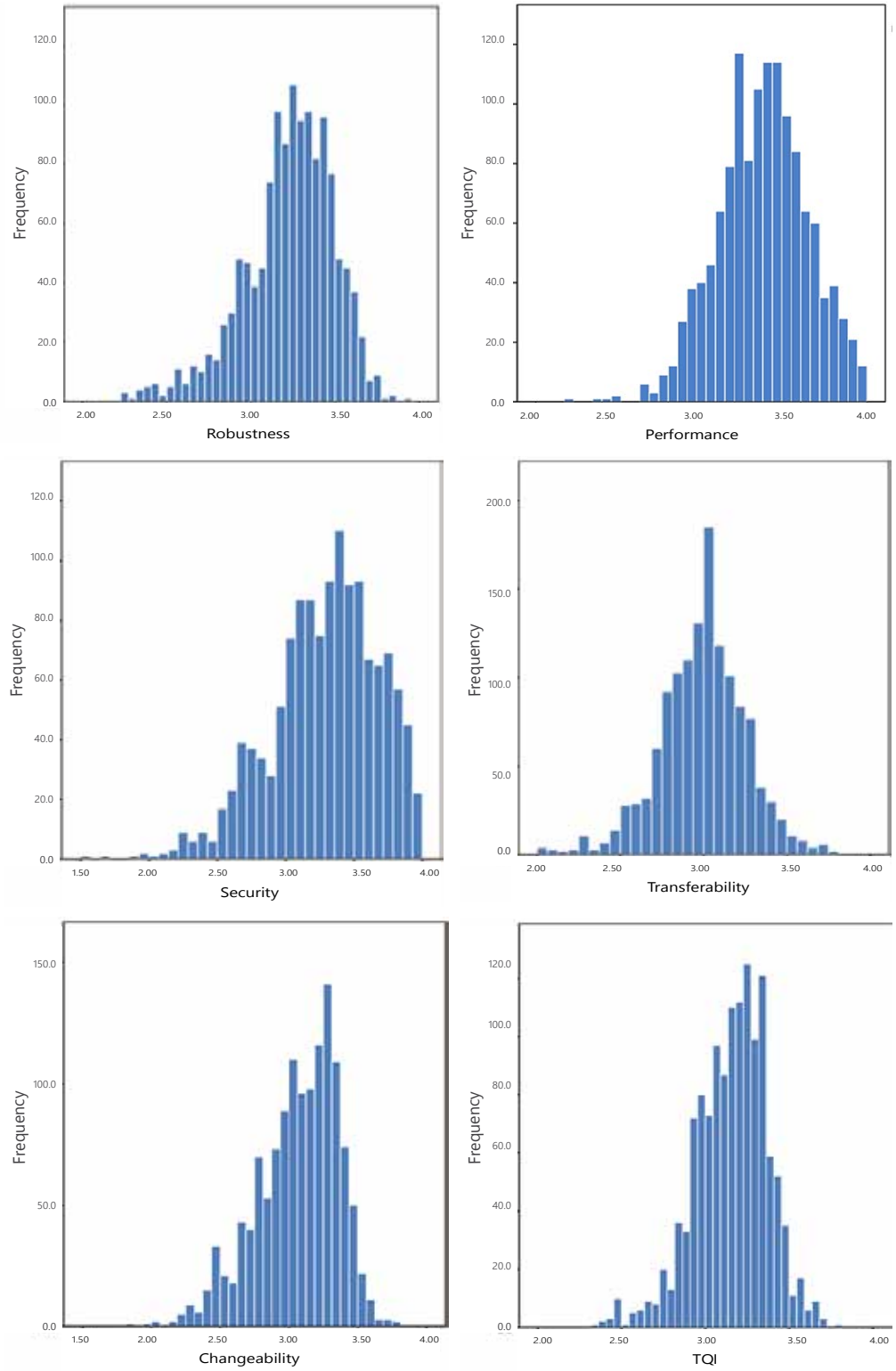


Table 3. Descriptive statistics for all Health Factors from the 2014 sample

Count	Mean	Std. Dev	Minimum	25th %	Median	75th %	Maximum
Robustness	3.23	0.27	2.23	3.09	3.26	3.41	3.94
Performance	3.21	0.34	1.74	2.99	3.22	3.44	3.99
Security	3.29	0.39	1.50	3.06	3.33	3.57	3.99
Transferability	2.99	0.26	2.01	2.83	3.01	3.16	3.81
Changeability	3.06	0.30	1.86	2.89	3.11	3.29	3.76
Total Quality	3.16	0.22	2.35	3.02	3.18	3.31	3.78

of its variance with any of them. Traditionally developers believed that the structural attributes which improved an application's performance would affect other quality attributes negatively. These results temper that belief, since Performance scores have weak relationships with other health factors.

The third observation, seen in the numbers in the first column of Table 4, is that the Total Quality Index is most heavily influenced by a combination of Robustness, Changeability, and Transferability. These three health factors share 56% to 72% of their variation with that of the Total Quality Index, while Performance and Security share only between 32% and 37% with it. Since Robustness, Changeability, and Transferability are strongly intercorrelated, they consequently have the largest influence on Total Quality Index scores.

The fourth and final observation is that the health factors have little to no relation to size when analyzed across the entire CRASH sample. The health factors share between 0% and at most 5% of their variation with size as measured in thousands of lines of code. However, the results differ among languages, with small relationships to size observed in Java-EE and COBOL as illustrated in Table 5. Table 5 presents the percent of shared variance between the five health factors and size for the Java-EE and COBOL samples. In the Java-EE sample, all health factors except Performance show

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	10		5	14	13	0
Security	37	36	5		2	7	2
Changeability	56	38	14	2		30	1
Transferability	61	34	13	7	30		1

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

a small relationship with size, the relationship for Robustness being the strongest. The scatterplot for Robustness with size is shown in Figure 3, where size is plotted on a logarithmic scale to make the relationship more visible. The underlying correlations are negative; indicating that as size grows the health factor scores decline.

For the COBOL sample, modest relationships with size are evident for all health factors except Transferability. Underlying correlations indicate that all of these relationships are negative with the exception of the modest relationship for Changeability which is positive. In COBOL, the relationship with size is strongest for Security. In Figure 4 size is presented on a logarithmic scale and shows that Security scores were above 3.5 for most COBOL applications up to about 3 million lines of code. Up to this point only a minority of applications had scores below 3.5, but after this point almost all scores fall below 3.5.

Table 5. Percent of shared variance among health factors and size for Java-EE and COBOL

	Count	Robust	Perform	Security	Change	Transfer
Java - EE	565	12	0	4	6	5
Cobol	280	6	2	16	7	0

All relationships are significant at $p < .001$

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

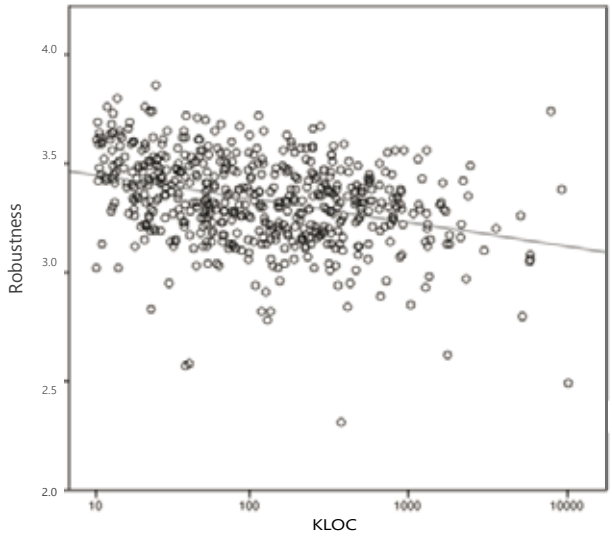
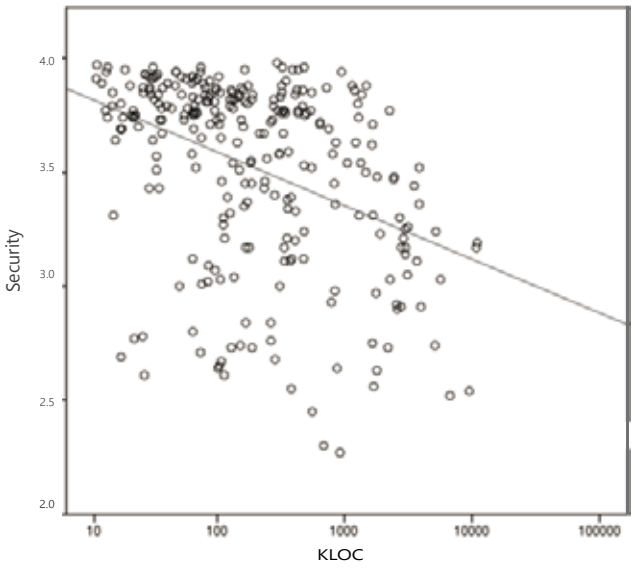


Figure 4. Scatterplot of Security scores with size in COBOL



3. Health Factor Scores by Language

This section will present box and whisker plots of score distributions along with the associated descriptive statistics for health factors in each language. The box and whisker plots present a quick visual representation of how scores are distributed across health factors in each language. The accompanying descriptive statistics can be used as rough benchmarks for evaluating the status of applications in each language analyzed using CAST's Application Intelligence Platform. These tables are best suited for benchmarking an application against its quartile in the Appmarq sample for a specific language.

3.1. Java-EE Health Factor Results

The descriptive statistics for each of the health factors and the Total Quality Index are presented for Java-EE in Table 6 and depicted as box and whisker charts in Figure 5. The scores for Transferability are significantly lower than those for the other health factors. Although it is difficult to directly compare health factor scores, the size of the disparity in scores for Transferability highlights an area for concern in Java-EE applications. At least seventy-five percent of scores for all health factors fall above 3.0, except Transferability, where barely more than 50% fall above 3.0. However, TQI and all health factors have outliers and extreme scores that fall below 2.5, scores that in practice have identified an area of trouble in applications.

The variation in Performance and Security is higher than for other health factors. It is difficult to pinpoint the exact causes of this variation. Even so, it is concerning to see such wide variation in health factors that affect the risk of an application to the business, especially in application security. Although Security exhibited the second highest mean and median scores among the health factors, it also exhibited the largest range with a long tail of negative scores characteristic of insecure applications.

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

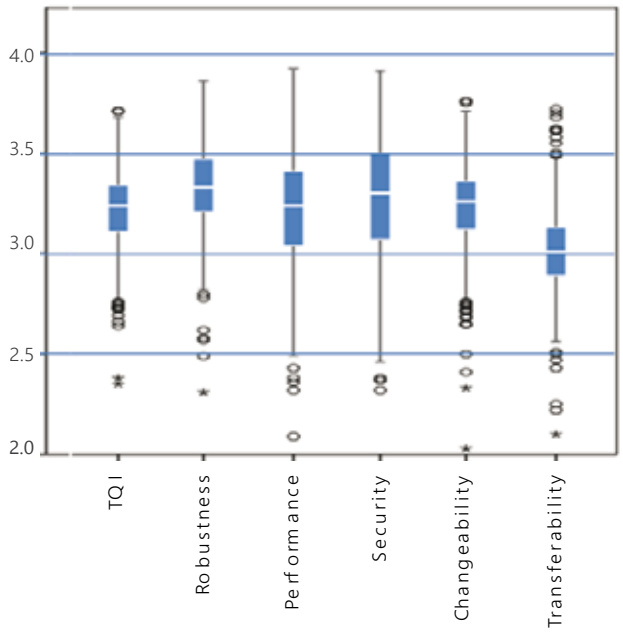


Table 6. Descriptive statistics by health factor for Java-EE applications

Statistic	TQI	Robust	Perform	Security	Change	Transfer
Sample	565	565	565	565	565	565
Mean	3.22	3.33	3.22	3.27	3.23	3.02
Std. Dev	.18	0.2	.29	.31	.20	.21
Maximum	3.71	3.86	3.92	3.91	3.76	3.72
75th %-ile	3.34	3.47	3.41	3.50	3.36	3.13
Median	3.24	3.33	3.24	3.30	3.26	3.01
25th %-ile	3.11	3.21	3.04	3.07	3.12	2.89
Minimum	2.35	2.31	1.74	1.50	2.03	2.10

3.2. COBOL Health Factor Results

The descriptive statistics for each of the health factors and the Total Quality Index are presented for COBOL in Table 7 and depicted as box and whisker charts in Figure 6. Security scores for COBOL applications are significantly higher than other health factor scores while Changeability scores are significantly lower. Since most of the COBOL applications are in either the financial services or insurance industry, the elevated Security scores represent the critical concern with protection of confidentiality and financial assets in these industries. Also, since COBOL runs on mainframes it is not as exposed to the security issues characteristic of applications that run on servers in an environment closely coupled to the internet.

The significantly lower Changeability scores reflect the complexity created when the larger module sizes in COBOL are continually modified over several decades to improve performance and security. In previous research we found that the average module size in COBOL was 600 lines of code, while in most modern languages it was 50 lines of code. Code units in Java-EE were even smaller, typically around 30 lines of code. Although higher than Changeability scores, Transferability scores fell below scores for Robustness, Performance, and Security. From these results it would appear that in environments where COBOL applications still run core business functions, IT has prioritized the reduction of operational risk over factors that increase the cost of maintenance.

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

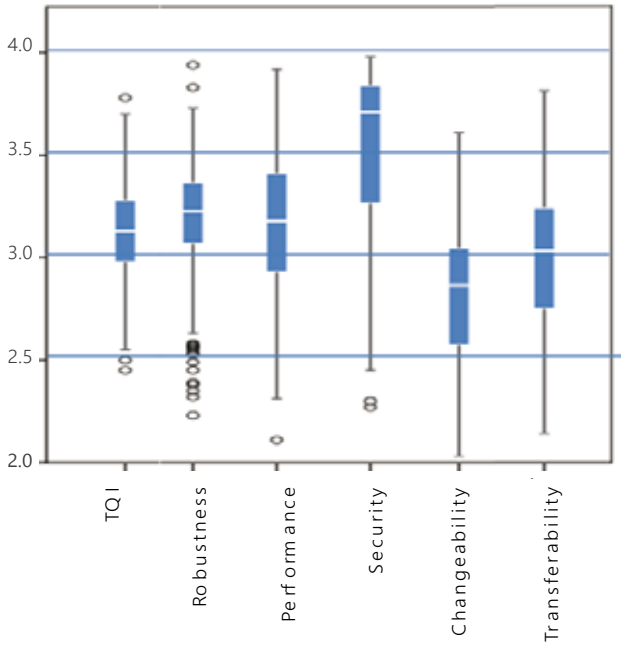


Table 7. Descriptive statistics by health factor for COBOL applications

Statistic	TQI	Robust	Perform	Security	Change	Transfer
Sample	280	280	279	280	280	280
Mean	3.13	3.19	3.17	3.53	2.83	3.01
Std. Dev	.23	.28	.35	.41	.31	.33
Maximum	3.78	3.94	3.92	3.98	3.61	3.81
75th %-ile	3.28	3.37	3.41	3.84	3.05	3.24
Median	3.13	3.23	3.28	3.71	2.87	3.03
25th %-ile	2.98	3.07	2.93	3.26	2.58	2.75
Minimum	2.45	2.23	2.11	2.27	2.03	2.14

3.3. .NET Health Factor Results

The descriptive statistics for each of the health factors and the Total Quality Index are presented for .NET in Table 8 and depicted as box and whisker charts in Figure 7.

Scores for Robustness were slightly higher than for other health factors. In a pattern similar to that seen with Java-EE, the variation in Performance and Security scores were greater than the variation for other health factors.

Figure 7. Box and whisker plots for TQI and health factors in .NET

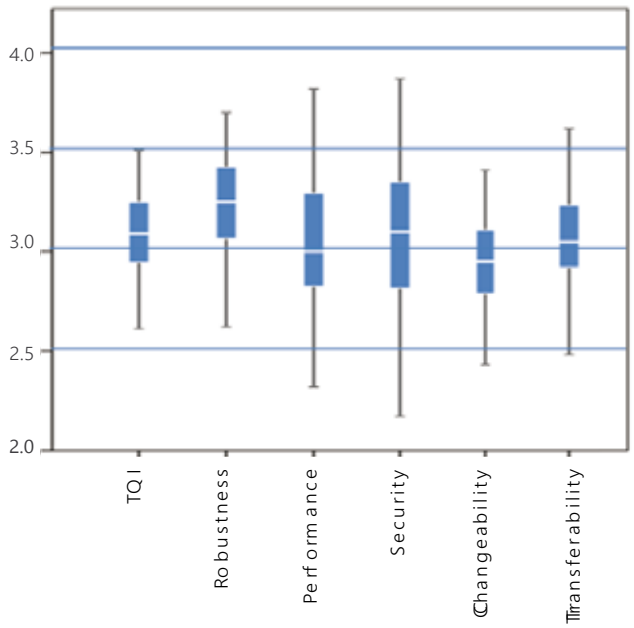


Table 8. Descriptive statistics by health factor for .NET applications

Statistic	TQI	Robust	Perform	Security	Change	Transfer
Sample	127	127	127	127	127	127
Mean	3.09	3.24	3.04	3.08	2.95	3.06
Std. Dev	.20	.23	.32	.36	.21	.22
Maximum	3.51	3.70	3.82	3.87	3.41	3.62
75th %-ile	3.25	3.43	3.30	3.35	3.11	3.24
Median	3.09	3.25	3.00	3.10	2.95	3.05
25th %-ile	2.94	3.06	2.82	2.81	2.79	2.92
Minimum	2.61	2.62	2.32	2.17	2.43	2.48

3.4. ABAP Health Factor Results

The descriptive statistics for each of the health factors and the Total Quality Index are presented for ABAP in Table 9 and depicted as box and whisker charts in Figure 8. ABAP is a language used for customizing applications built atop the SAP application platform. The scores for Security were substantially higher than those for other health factors. These elevated scores may reflect that SAP applications often involve financial information that places a priority on security. The largest variation was observed for Performance scores. Performance is frequently cited as challenging in packaged applications that have been customized using ABAP.

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

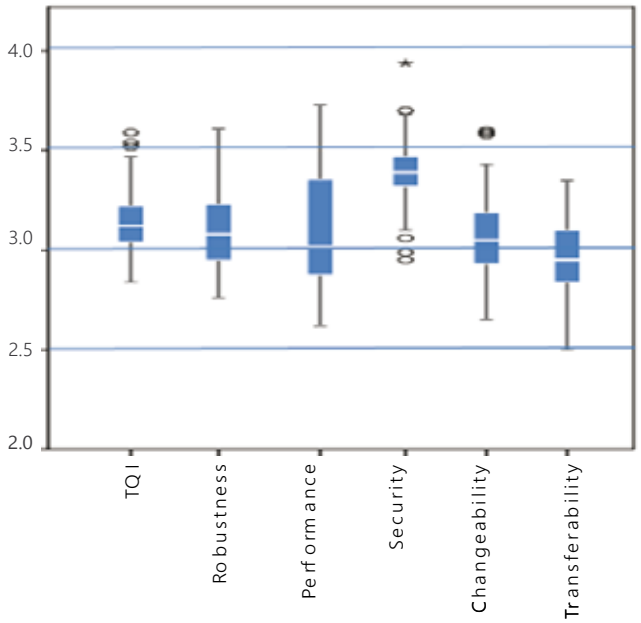


Table 9. Descriptive statistics by health factor for ABAP applications

Statistic	TQI	Robust	Perform	Security	Change	Transfer
Sample	77	77	76	77	77	77
Mean	3.15	3.10	3.09	3.40	3.07	2.97
Std. Dev	.17	.19	.28	.17	.21	.20
Maximum	3.59	3.61	3.73	3.94	3.60	3.35
75th %-ile	3.22	3.23	3.36	3.47	3.19	3.10
Median	3.12	3.08	3.02	3.39	3.05	2.95
25th %-ile	3.04	2.95	2.88	3.32	2.93	2.84
Minimum	2.84	2.76	2.62	2.95	2.65	2.50

3.5. Oracle Forms Health Factor Results

The descriptive statistics for each of the health factors and the Total Quality Index are presented for Oracle Forms in Table 10 and depicted as box and whisker charts in Figure 9. Oracle Forms is a language used for customizing applications built atop the Oracle application platform. On average, Changeability and especially Transferability scores were lower than for the other health factors. Performance scores were generally higher. Although the median for Performance and Security scores were very close, the greater variability in the lower tail of Security scores resulted in a lower mean.

Figure 9. Box and whisker plots for TQI and health factors in Oracle Forms

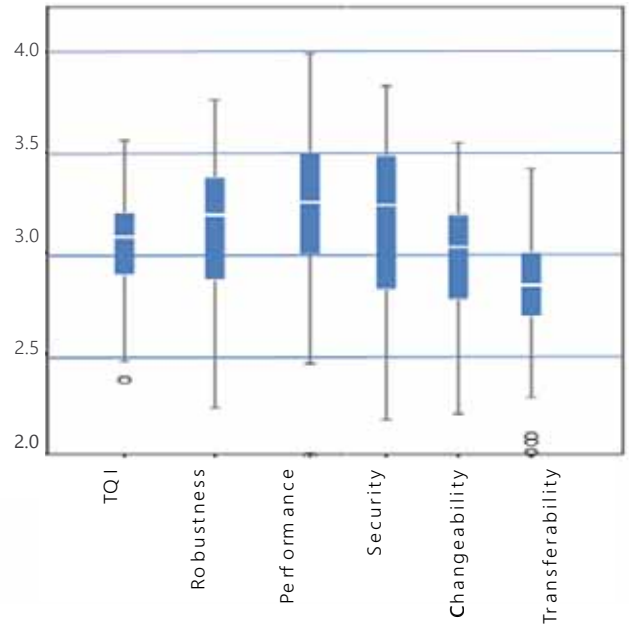


Table 10. Descriptive statistics by health factor for Oracle Forms applications

Statistic	TQI	Robust	Perform	Security	Change	Transfer
Sample	59	59	59	59	59	59
Mean	3.02	3.09	3.22	3.15	2.94	2.81
Std. Dev	.29	.37	.39	.45	.38	.30
Maximum	3.56	3.76	3.99	3.83	3.55	3.42
75th %-ile	3.20	3.38	3.51	3.49	3.19	3.01
Median	3.08	3.19	3.25	3.24	3.03	2.84
25th %-ile	2.97	2.86	2.99	2.82	2.77	2.66
Minimum	2.37	2.23	1.99	1.71	1.86	2.01

3.6. Oracle ERP/CRM Health Factor Results

The descriptive statistics for each of the health factors and the Total Quality Index are presented for Oracle ERP/CRM in Table 11 and depicted as box and whisker charts in Figure 10. Performance scores for Oracle ERP/CRM applications are substantially higher than scores for the other health factors. The median scores for both Changeability and Transferability were lower than scores for the other health factors. The variability for both Performance and Security was greater than for the other health factors. In particular, the interquartile range for Security extended almost to the lower range of scores, indicating a dense collection of applications at the bottom of the Security distribution.

Figure 10. Box and whisker plots for TQI and health factors in Oracle ERP/CRM

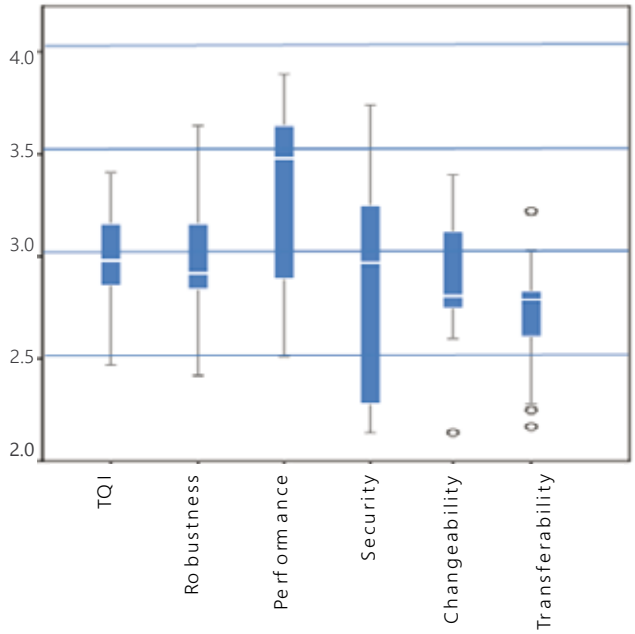


Table 11. Descriptive statistics by health factor for Oracle ERP/CRM applications

Statistic	TQI	Robust	Perform	Security	Change	Transfer
Sample	33	33	29	29	33	33
Mean	2.97	2.98	3.30	2.86	2.89	2.71
Std. Dev	.24	.26	.42	.51	.25	.24
Maximum	3.41	3.64	3.89	3.74	3.40	3.22
75th %-ile	3.16	3.16	3.64	3.25	3.12	2.83
Median	2.98	2.92	3.48	2.97	2.81	2.79
25th %-ile	2.86	2.84	2.89	2.28	2.75	2.61
Minimum	2.47	2.42	2.51	2.14	2.14	2.17

3.7. C Health Factor Results

The descriptive statistics for each of the health factors and the Total Quality Index are presented for C in Table 12 and depicted as box and whisker charts in Figure 11. Performance and Security scores were substantially higher than those for other health factors. C is often chosen for performance-sensitive applications since it allows developers to gain greater control over the machine. Scores for Changeability and Transferability were the lowest, which is not surprising since the greater access a language provides to the machine running it, the more complex a program often becomes as developers modify the code to optimize specific aspects of machine performance.

Figure 11. Box and whisker plots for TQI and health factors in C

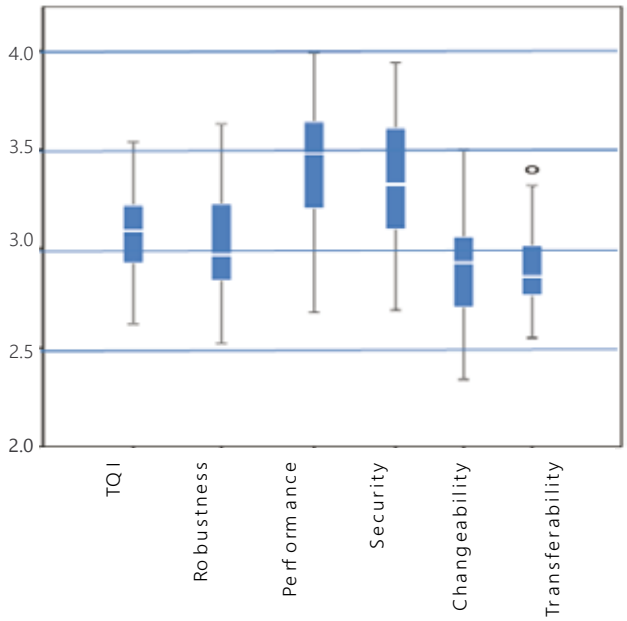


Table 12. Descriptive statistics by health factor for C applications

Statistic	TQI	Robust	Perform	Security	Change	Transfer
Sample	39	39	36	36	39	39
Mean	3.09	3.01	3.41	3.32	2.91	2.90
Std. Dev	.22	.27	.33	.33	.27	.23
Maximum	3.54	3.63	3.99	3.94	3.50	3.40
75th %-ile	3.22	3.25	3.64	3.61	3.06	3.02
Median	3.09	2.97	3.48	3.33	2.93	2.86
25th %-ile	2.93	2.84	3.21	3.10	2.66	2.76
Minimum	2.62	2.52	2.68	2.69	2.34	2.55

3.8. C++ Health Factor Results

The descriptive statistics for each of the health factors and the Total Quality Index are presented for C++ in Table 13 and depicted as box and whisker charts in Figure 12. The pattern of scores for C++ is very similar to the pattern for C. The scores for Security and Performance are substantially higher than scores for the other health factors, while scores for Changeability and Transferability were the lowest. C++ shares many of the machine-accessible attributes of C for performance-sensitive applications.

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

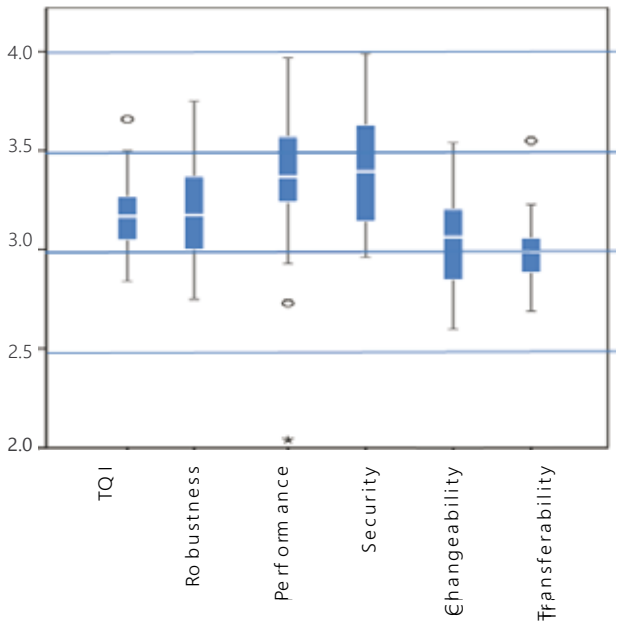


Table 13. Descriptive statistics by health factor for C++ applications

Statistic	TQI	Robust	Perform	Security	Change	Transfer
Sample	28	28	25	28	28	28
Mean	3.18	3.19	3.36	3.42	3.03	2.98
Std. Dev	.19	.26	.39	.30	.24	.19
Maximum	3.66	3.75	3.97	3.99	3.54	3.55
75th %-ile	3.27	3.37	3.57	3.63	3.21	3.06
Median	3.17	3.18	3.37	3.40	3.07	2.99
25th %-ile	3.05	3.00	3.24	3.15	2.85	2.89
Minimum	2.84	2.75	2.04	2.96	2.60	2.69

3.9. ASP Health Factor Results

The descriptive statistics for each of the health factors and the Total Quality Index are presented for ASP in Table 14 and depicted as box and whisker charts in Figure 13. ASP exhibits a heath factor pattern similar to that for C and C++ with scores for Performance and Security higher than those for the other health factors, while scores for Changeability and Transferability are lower. The variation in scores for Performance and Transferability was greater than for other health factors.

Figure 13. Box and whisker plots for TQI and health factors in ASP

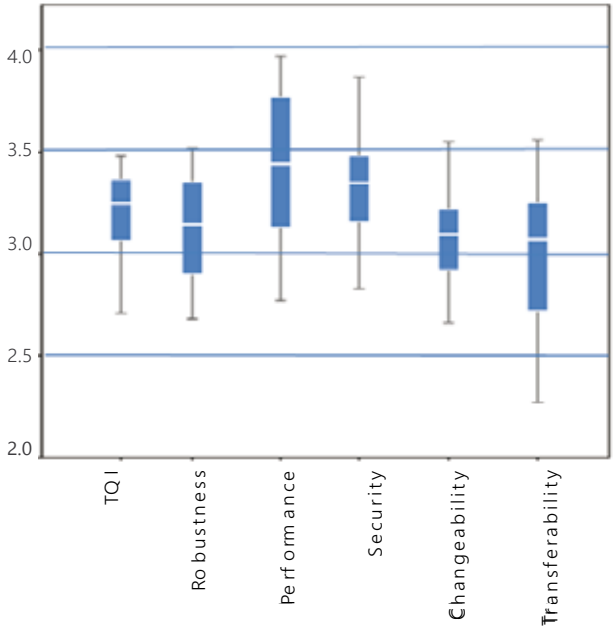


Figure 14. Box and whisker plots for TQI and health factors in ASP

Statistic	TQI	Robust	Perform	Security	Change	Transfer
Sample	24	24	21	24	24	24
Mean	3.20	3.13	3.44	3.33	3.08	3.00
Std. Dev	.20	.26	.37	.25	.24	.35
Maximum	3.48	3.52	3.97	3.87	3.55	3.56
75th %-ile	3.37	3.35	3.77	3.48	3.22	3.26
Median	3.25	3.15	3.44	3.35	3.10	3.07
25th %-ile	3.07	2.90	3.13	3.16	2.92	2.72
Minimum	2.71	2.68	2.77	2.83	2.66	2.27

3.10. Summary of Health Factor

Several patterns emerge across languages for the health factor scores. In both COBOL and ABAP the scores for Security were elevated above those for the other health factors, perhaps reflecting the greater tendency for applications written in these languages to manage financial information. In C, C++, Oracle Forms, and ASP the scores for both Performance and Security are elevated above the other health factors, while scores for Changeability and Transferability were lower. Scores for Changeability and Transferability were also lower in Oracle ERP/CRM, but scores for Performance were substantially higher than those for other health factors. The most discernable pattern in Java-EE was the substantially lower scores for Transferability. Across the languages, Performance and Security most often displayed the greatest variability in scores.

4. Impact of Demographic Factors on Structural Quality

This section will present health factor scores across the various categories of demographic variables reported for applications in the CRASH sample. Demographic information was not reported for all applications in the sample. As a result, only the large sample of Java-EE applications contains a sufficient number of applications in each category of the demographic variables to make statistically valid inferences from the data. For a couple of demographic variables there is sufficient data across the various categories in COBOL applications to supplement the analyses reported for Java-EE.

Differences in health factor scores for the various categories of the demographic variables reported here were all tested statistically using the F-Test for differences between mean scores. The criterion for significance was set at $p = .05$, a typical level for testing the significance of results in scientific research. In essence, the observed differences between means for the various categories could not occur by chance more than once in 20 times for the result to be significant.

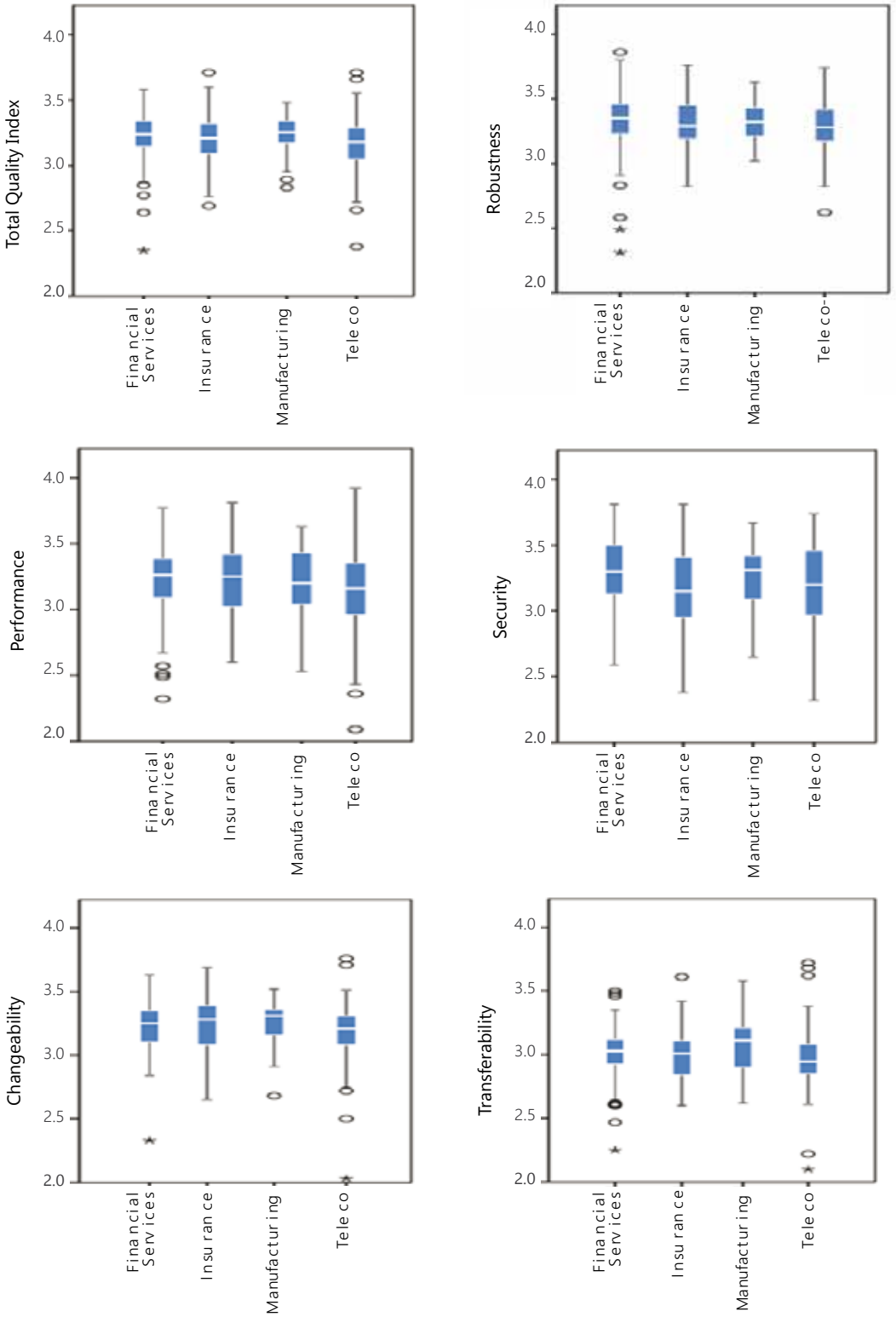
For samples as large as the number of the Java-EE applications in the CRASH data, a result can achieve significance even though its impact is small in terms of the total percent of variation among the scores that it affects. In order to assess a demographic variable's impact, for each significant result we report the percent of the variation among scores that are accounted for by differences among categories of the demographic variable. For demographic variables with more than two categories we also report post hoc analyses that indicate which categories were responsible for producing the statistically significant differences observed.

4.1. Industry Sectors

Analyzing health factor scores across industry sectors is difficult since some industry sectors differ on their use of languages. Consequently, comparisons between industry sectors, especially when they involve financial services and manufacturing, may be strongly influenced by differences between languages. Structural quality differences among industry sectors may represent differences in their inherent characteristics. However, they could just as easily result from differences in the mix of languages used in the applications sampled in industry sectors. Therefore the most valid comparisons among industry sectors are those made within a single language or technology. The most widely used language across industry sectors in the CRASH sample is Java-EE. The following analysis reports differences between industry sectors that contributed at least 60 Java-EE applications to the 2014 CRASH sample.

Figure 14 displays the health factor results for financial services (179 applications), telecommunications (106 applications), manufacturing (65 applications), and insurance (61 applications). Significant but very small differences were observed between industry sectors for Security, Changeability, and Transferability scores. Differences among industry sectors for Security ($F = 3.16$, $df = 3, 407$, $p < .03$), Transferability ($F = 2.98$, $df = 3, 407$, $p < .04$), and Changeability (F

Figure 14. Health factor distributions by industry sector



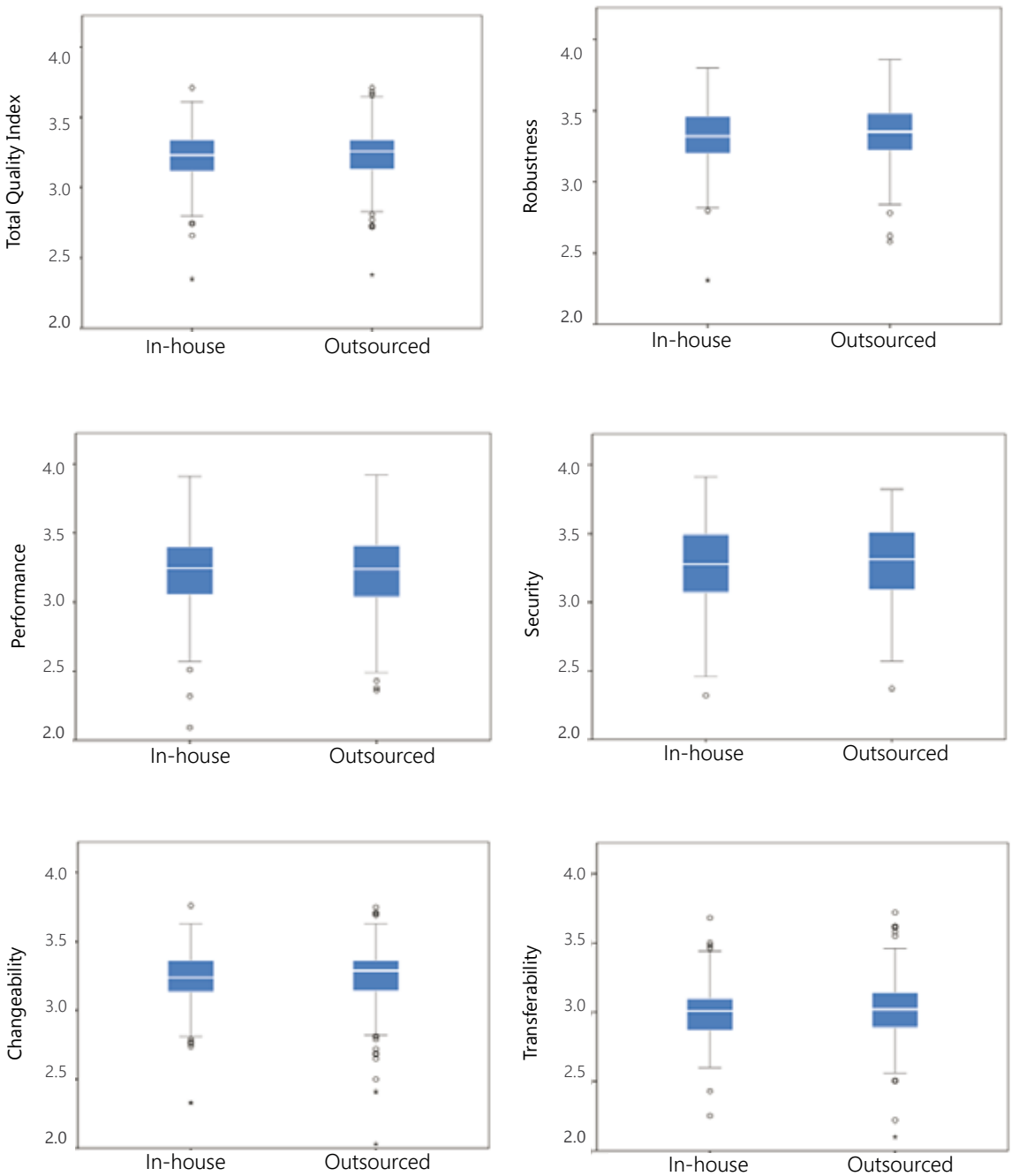
= 2.85, df = 3, 407, p < .04) each accounted for only 2% of the variation among scores. These results suggest that differences among industrial sectors are small, and that factors other than industrial domain have more impact on the structural quality of applications.

4.2. Source

The sourcing of applications was divided between in-house and outsourced. Figure 15 displays the distributions of scores for in-house and outsourced applications for each health factor. Of the 501 Java-EE applications that reported sourcing information, 224 were developed in-house, while 277 were outsourced. There were no significant differences in the average size measured in lines of code between in-house and outsourced applications.

As if evident visually from this figure and was confirmed statistically, there were no significant differences between sourcing choices on any health factor scores in the Java-EE sample. Although there were no mean differences on these health factors, there was substantial variation within each sourcing category suggesting that factors other than application source might be affecting the scores.

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

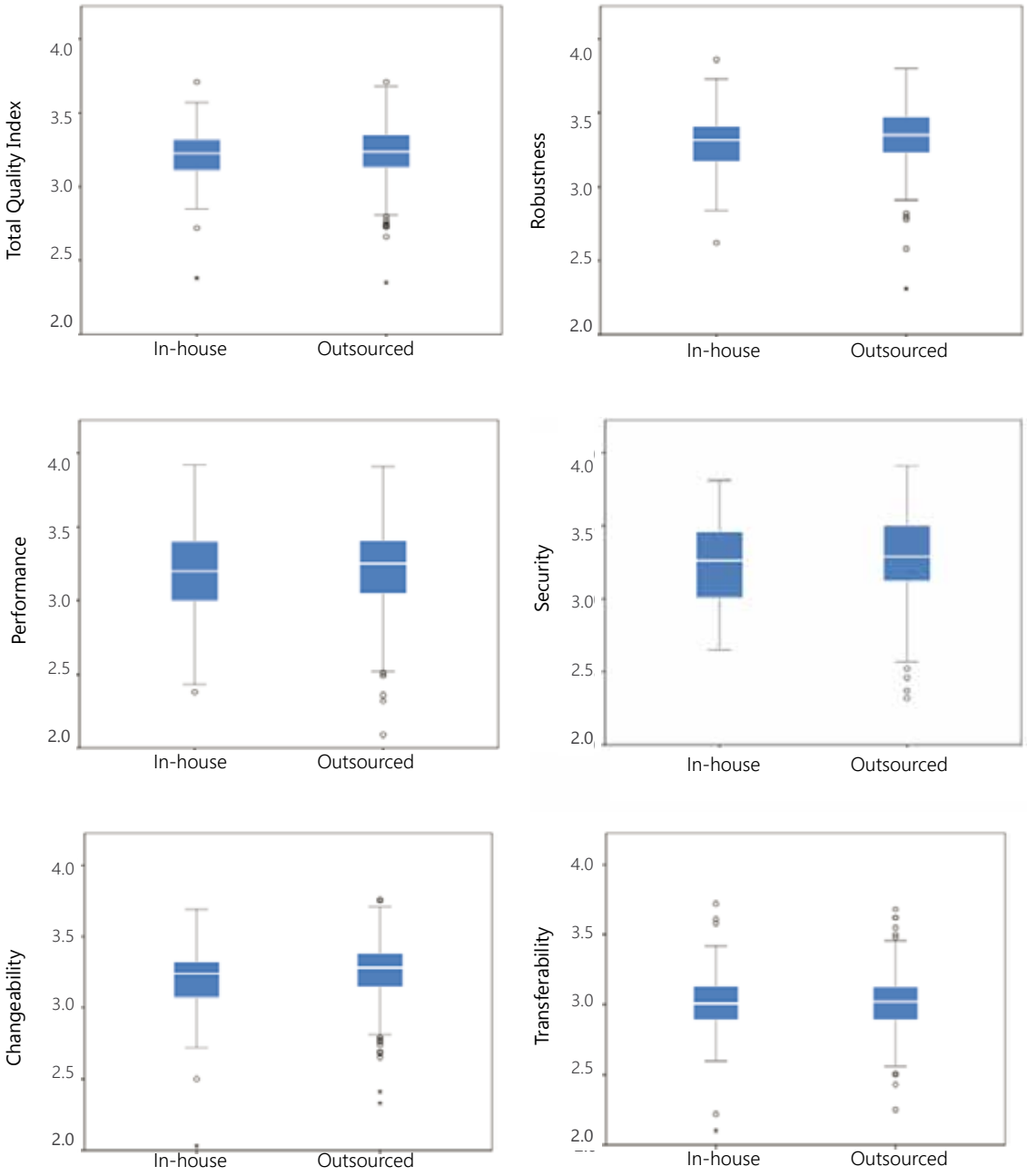


4.3. Shore

In the Java-EE sample, 387 applications were developed onshore while 114 were developed or maintained offshore. Figure 16 displays the distributions of scores for onshore and offshore applications for each health factor. There were no statistically significant differences in scores for Performance, Security, and Transferability between applications developed onshore or offshore. There were no significant differences in the average size measured in lines of code between onshore and offshore applications.

The most significant difference based on shoring choice, although small, was observed on Changeability scores ($F = 8.78$, $df = 1, 499$, $p < .005$). Onshore applications were slightly more changeable, but this difference accounted for less than 2% of the variation in Changeability scores and is at appears to be a minor factor affecting the changeability of an application. Differences in Robustness scores also achieved statistical significance ($F = 5.91$, $df = 1, 499$, $p < .02$). However, this result only accounted for 1% of the variation among scores and is also only a minor factor affecting Robustness scores. As a result of differences in Changeability and Robustness scores achieving statistical significance, the scores for the Total Quality Index crossed the threshold for significance ($F = 4.05$, $df = 1, 499$, $p < .05$), although it accounts for less than 1% of the variation in scores.

Figure 16. Health factor distributions for onshore and offshore applications



4.4. CMMI Maturity Level

In the Java-EE sample, 23 applications were developed in CMMI Level 1 organizations, 26 were developed in CMMI Level 2 organizations, and 32 were developed in CMMI Level 3 organizations. There were not enough CMMI Level 4 or Level 5 organizations in the sample to allow a comparison beyond CMMI Level 3.

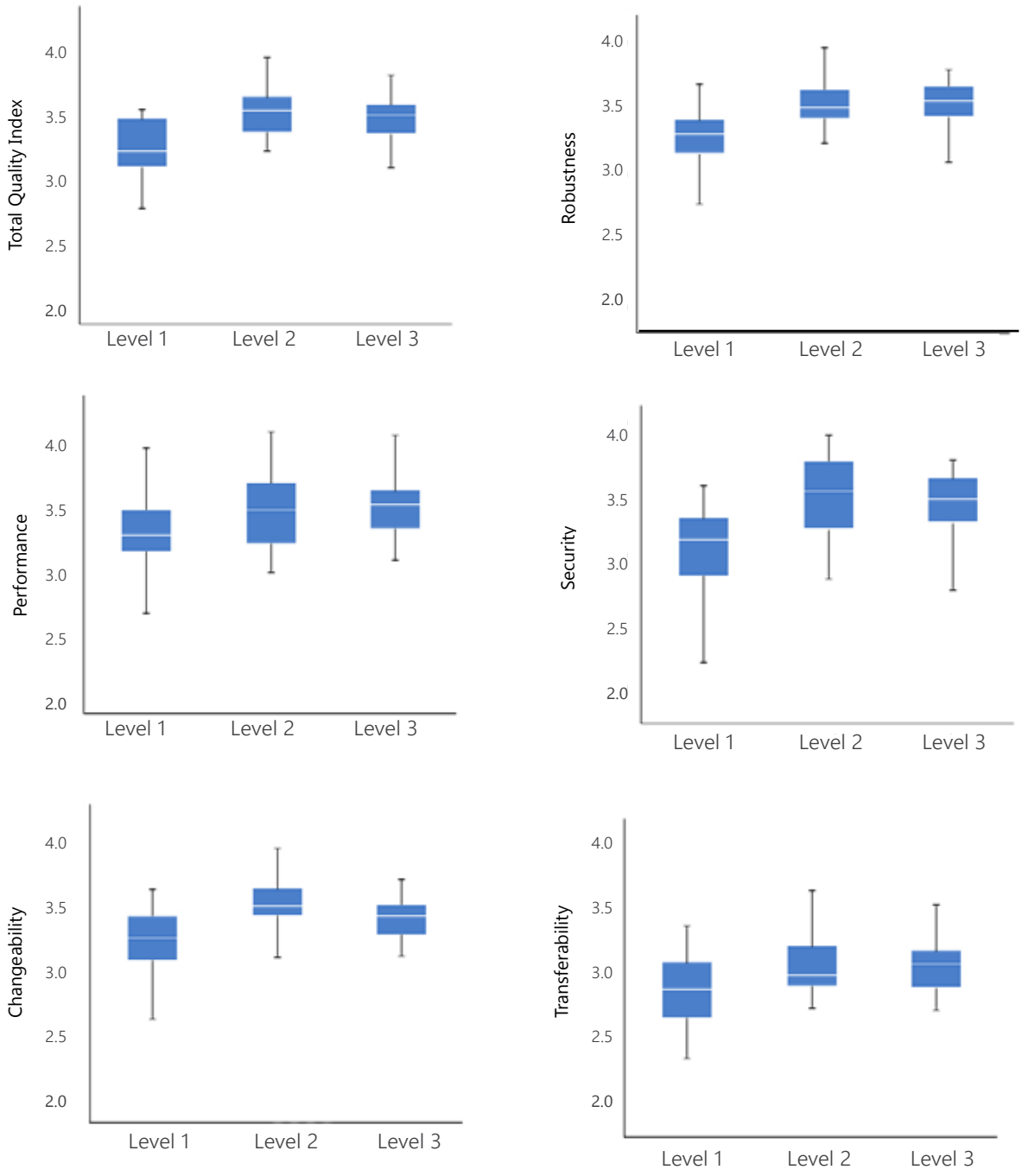
Figure 17 displays the distributions of scores for applications developed in CMMI Levels 1, 2, and 3 organizations for each health factor. Significant differences were observed among applications developed at different CMMI maturity levels on all health factors. There were no significant differences in the sizes as measured in lines of code between the applications developed in organizations at any of the three CMMI levels.

The strongest effects were observed for Robustness, Security, and Changeability. The significant result for Robustness ($F = 14.87$, $df = 2, 78$, $p < .001$) accounted for 28% of the variation in scores. The significant result for Security ($F = 13.15$, $df = 2, 78$, $p < .001$) accounted for 25% of the variation in scores. The significant result for Changeability ($F = 9.99$, $df = 2, 78$, $p < .001$) accounted for 20% of the variation in scores. The impact of CMMI Maturity Level on Performance and Transferability was not as strong as it had been on the other health factors, although as can be seen in Figure 16, the impact is substantial. The significant result for Performance was the weakest among the health factors ($F = 4.77$, $df = 2, 78$, $p < .02$), accounting for 11% of the variation in scores. The significant result for Transferability ($F = 5.12$, $df = 2, 78$, $p < .01$) accounted for 12% of the variation in scores.

Post hoc analyses confirmed that the significant mean differences observed on each health factor resulted from applications developed by Level 1 organizations having significantly lower structural quality scores that those developed in CMMI Level 2 or Level 3 organizations. Post hoc tests did not reveal any significant differences in average scores between CMMI Level 2 and Level 3 organizations on scores for any of the health factors.

These results are not surprising since the change from CMMI Level 1 to Level 2 involves removing some of the most common impediments to successful software engineering practice such as unachievable commitments and volatile requirements. With these problems under control developers are able to perform their work in a more orderly and professional manner resulting in fewer mistakes and earlier detection of those that are made. This improvment will have significant impact on the structural quality of the software. The change from CMMI Level 2 to Level 3 is focused more on achieving an economy of scale from standardizing development practices, so it is not surprising that structural quality scores were similar between the two levels. Nevertheless, these data offer definitive proof that process improvements can have strong positive effects on the structural quality of IT applications.

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



4.5. Development Method

In the Java-EE sample, 57 applications reported using Agile methods, 60 applications reported using Waterfall methods, 46 applications reported using a hybrid mix of Agile and Waterfall methods, and 21 projects reported using no method. Thirty-six applications reported using other methods for which there were insufficient numbers to create a separate category and will not be analyzed here since it is difficult to characterize the methods used with these applications.

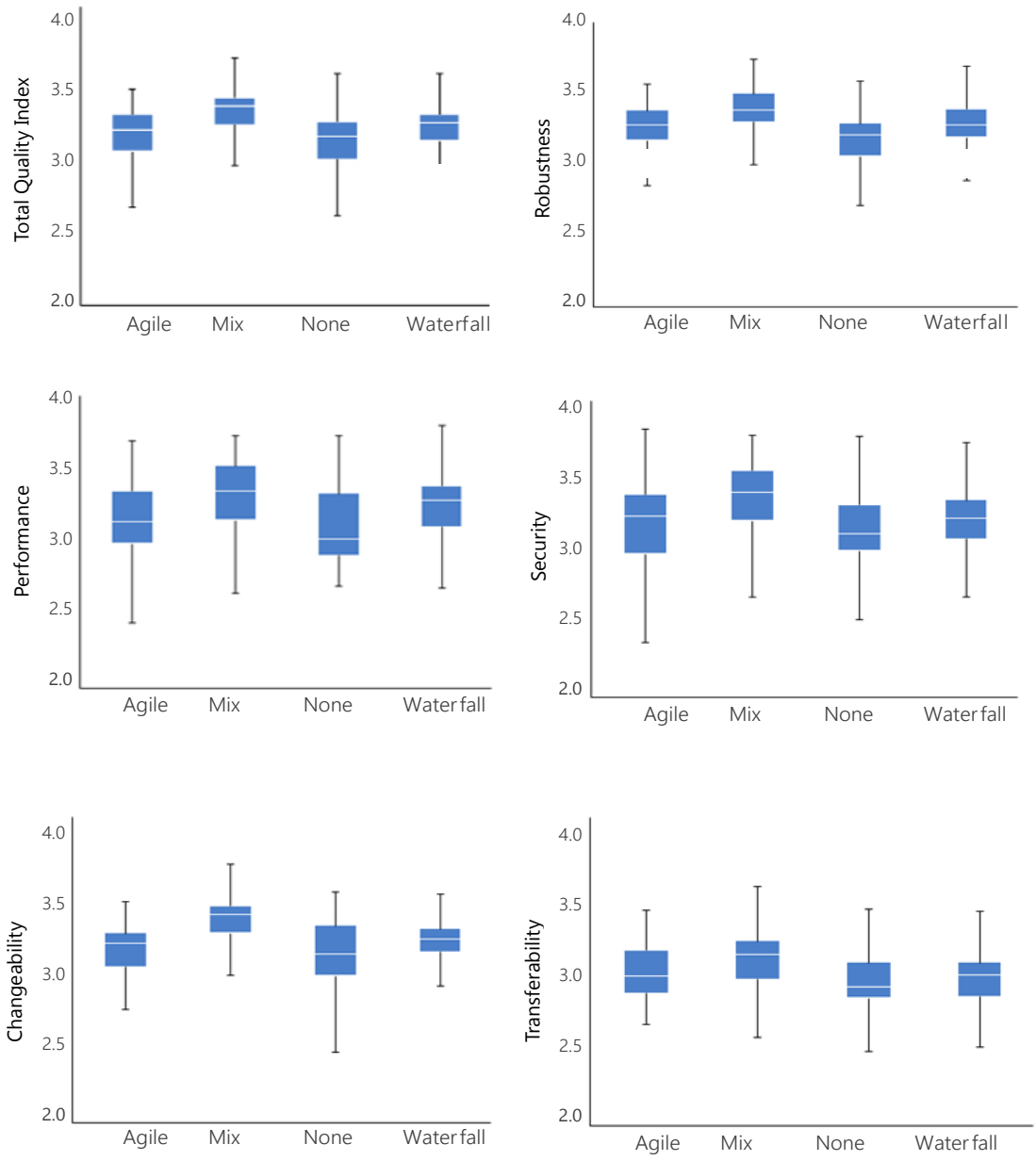
Figure 18 displays the distributions of scores for applications using different development methods. Significant differences were observed among development methods on all health factors. The strongest differences between development methods were observed for Robustness where they accounted for 15% of the variation in scores ($F = 9.14$, $df = 4, 214$, $p < .001$) and for Changeability where they accounted for 14% of the variation in scores ($F = 8.99$, $df = 4, 215$, $p < .001$). For both health factors, post hoc tests (Tukey HSD) confirmed that the significant differences were accounted for by the superiority of the hybrid mix of Agile and Waterfall methods compared to scores for the other methods. The other development methods did not differ significantly from each other.

Differences between development methods accounted for 9% of the variance in Security scores ($F = 5.404$, $df = 4, 215$, $p < .001$). Post hoc tests revealed that these differences were accounted for by the superiority of scores for the hybrid mix of Agile and Waterfall over scores for Agile methods or no method. Security scores for Waterfall methods were in the middle and not significantly different from scores for other methods.

The smallest differences between development methods were observed for Performance where they accounted for 6% of the variation in scores ($F = 3.624$, $df = 4, 215$, $p < .01$) and for Transferability where they accounted for 5% of the variation in scores ($F = 2.751$, $df = 4, 215$, $p < .03$). Post hoc tests (Tukey HSD) revealed that the Performance differences were accounted for by the superiority of scores for the hybrid mix of Agile and Waterfall over the scores for Agile methods, and that the Transferability scores were due to the superiority of scores for the mix of Agile and Waterfall over no method.

These results confirm that for the large business critical applications that compose the CRASH sample, the hybrid mix of Agile and Waterfall methods produces greater structural quality than other development methods, although for Performance and Transferability these differences are not large. The superiority of the hybrid Agile/Waterfall mix suggests that for these types of applications the greater emphasis on up front design leads to better structural quality results for the Robustness, Changeability, and Security of the application, and to a smaller extent for their Performance and Transferability.

Figure 18. Health factor distributions for development methods



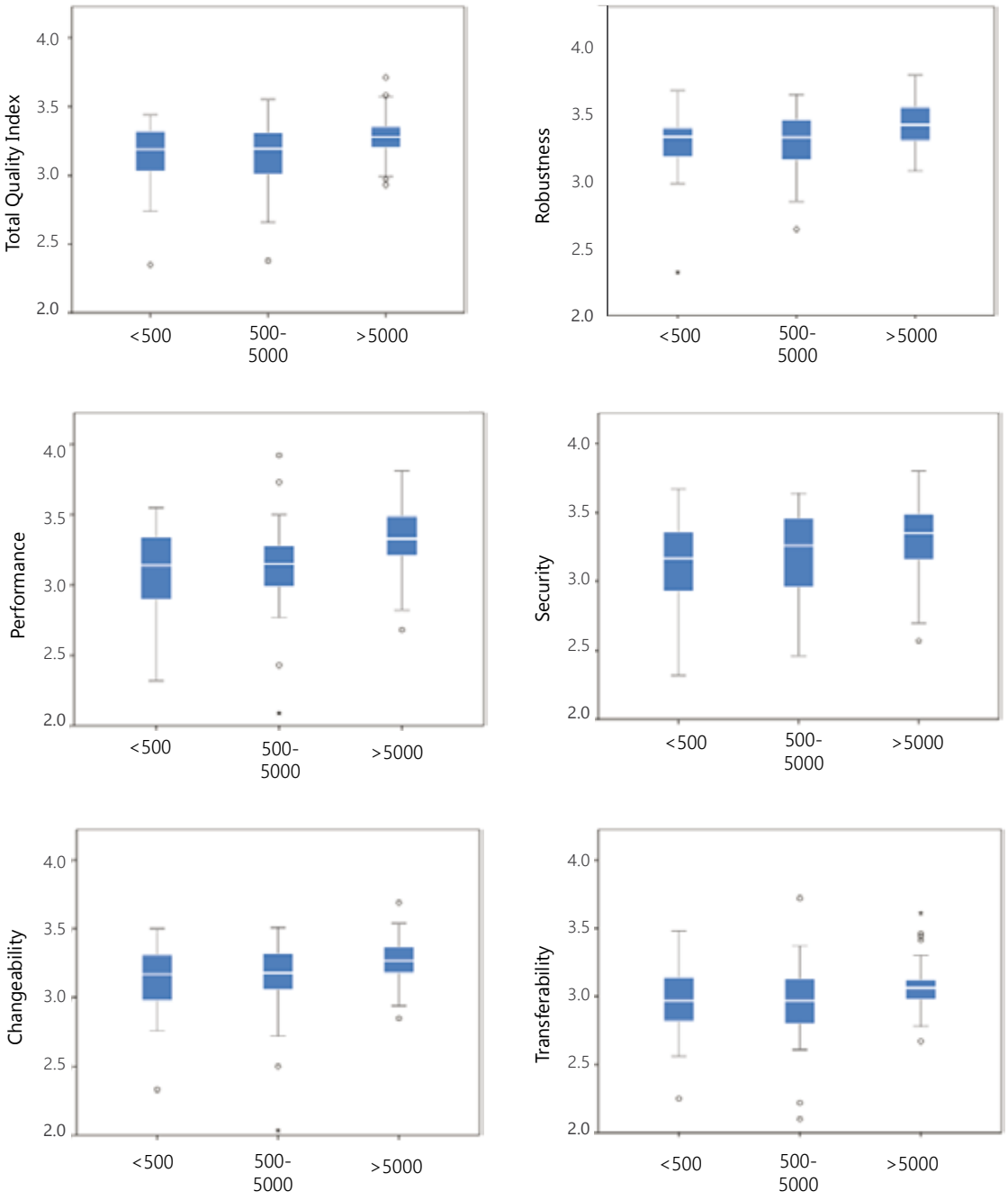
4.6. Number of Users

In the Java-EE sample 50 applications were reported to serve under 500 users, 37 applications served 500 to 5000 users, and 101 applications served more than 5000 users. Significant differences were found for all health factors based on the number of users served by the application. Figure 19 displays the distributions of scores for applications serving different numbers of users.

The strongest differences were observed for Performance ($F = 15.23$, $df = 2, 185$, $p < .001$), where differences in the number of users accounted for 14% of the variation among scores. Strong differences were also observed for Robustness ($F = 11.60$, $df = 2, 185$, $p < .001$), for Security ($F = 9.42$, $df = 2, 185$, $p < .001$), and for Changeability ($F = 8.12$, $df = 2, 185$, $p < .001$), where differences in the number of users accounted for 11%, 9%, and 8% of the variation among scores respectively. The weakest differences were observed for Transferability ($F = 4.06$, $df = 2, 185$, $p < .02$), where differences in the number of users accounted for 4% of the variation among scores.

For all the health factors, the significant differences were accounted for by the scores for applications serving more than 5000 users being higher than scores for those serving 5000 or fewer users. Applications serving more than 5000 users are typically customer facing applications and it is not surprising that 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.

Figure 19. Health factor distributions for number of end users



4.7. Summary of Demographic Effects on Health Factor Scores

The strongest effects among all the demographic variables measured were for process maturity wherein CMMI Level 1 organizations produced applications with substantially lower scores on all health factors than applications developed in CMMI Level 2 or Level 3 organizations. While process maturity accounted for 11% or more of the variation among all health factor scores, the impact on Robustness and Security was especially strong, accounting for at least a quarter of the variation.

The impact of development method was not as great as that for process maturity, but still impacted all health factor results accounting for between 5% and 15% of the variation in scores. Across all health factors, a hybrid mix of Agile and Waterfall methods produced higher scores than either Agile of Waterfall methods. These results suggest that for business critical applications the value of agile and iterative methods is enhanced by the types of up-front architectural and design activity that characterized Waterfall methods.

The number of users served by an application had similar sized effects to those for development methods, accounting for between 4% and 14% of the variation among health factor scores. Not surprisingly, applications serving more than 5000 users had higher scores on all health factors likely because they were typically customer-facing.

The industry sector in which an application was developed had a small effect on its Security, Changeability, and Transferability. The choice to develop applications in-house versus outsourcing had no effect on structural quality scores. The choice of developing applications onshore versus offshore had significant although minimal effects on Changeability and Robustness. These factors were not as important as process maturity, development method, or the number of users in affecting the structural quality of applications.

These results provide definitive evidence for the value of process maturity and a hybrid mix of agile and iterative methods in developing business critical applications. In the sample structural quality on large business critical applications was best achieved when impediments to disciplined software engineering practices were removed and early design activity was integrated with short cycle releases. Process maturity and development method were more important than where or by which organization the application was developed.

.

Authors

Dr. Bill Curtis
Senior Vice President and Chief Scientist

Dr. Bill Curtis is best known for leading development of the Capability Maturity Model (CMM). Prior to joining CAST, Bill was a Co-Founder of TeraQuest, the global leader in CMM-based services. Earlier he directed the Software Process Program at the Software Engineering Institute (SEI) at Carnegie Mellon University. He also directed research at MCC, at ITT's Programming Technology Center, in GE Space Division, and at the University of Washington. He is a Fellow of the Institute of Electrical and Electronics Engineers for his contributions to software process improvement and measurement.

Lev Lesokhin
Executive Vice President, Strategy and Analytics

Lev Lesokhin is responsible for CAST's market development, strategy, thought leadership and product marketing. He has a passion for customer success, building the ecosystem, and advancing the state of the art in business technology. Lev comes to CAST from SAP's Global SME organization. Prior to SAP, Lev was a leader in the research team at the Corporate Executive Board, a consultant at McKinsey and a member of technical staff at the MITRE corporation. Lev holds an MBA from the Sloan School of Management at MIT and a B.S.E.E. from Rensselaer Polytechnic Institute.

Alexandra Szynkarski
Product Marketing Manager

Alexandra Szynkarski is the product manager for CAST Highlight and research assistant in CAST Research Labs. Her research interests include comparative analysis of application technical quality across technologies and industry verticals, as well as measuring technical debt. Alexandra received an MS in international business administration from the Institut d'Administration des Entreprises in Nice, France.

Stanislas Duthoit
Research Associate

Stanislas Duthoit is a research associate in CAST Research Labs. His interests include structural quality benchmarks and measuring software performance trends on the global application development community. Stanislas holds a MSc in Civil Systems and a Certificate in Management of Technology from UC Berkeley.

CAST Research Labs

CAST Research Labs (CRL) champions the empirical study of software implementation in business technology. We provide practical advice and biennial benchmarks to the global application development community, as well as interacting with the academic community. Through scientific analysis of large software applications, we focus on providing insights that can improve application structural quality.

Starting in 2007, CRL has been collecting metrics and structural characteristics from custom applications deployed by large, IT-intensive enterprises across North America, Europe, and India. This unique dataset, stored in the CAST Appmarq benchmarking repository, currently stands at approximately 1,300 applications (representing over 700 million lines of code) and forms a basis to analyze actual software implementation in industry. As a baseline, CRL publishes a biennial CRASH Report of software trends across technologies and industry sectors found in our Appmarq repository.

For more information, please visit: http://www.castsoftware.com/Research-Labs