UDC 519.816

RESEARCH VERSUS PRACTICE IN SOFTWARE ENGINEERING: COMPARISON OF EXPERT OPINIONS TO MEASURED USER PRIORITIES

M. HAIGH

This work explores the differences in software quality perceptions between different groups of people involved with the software development process. Three hundred and fifteen respondents ranked each of thirteen generally accepted attributes of software quality on a scale of one to seven according to their perceived importance for the piece of software most vital to that individual's work. Differences in the priorities assigned to these attributes were explored using a number of different statistical techniques. Results of this research were compared to the results of several existing studies conducted by experts in theory and practice of software engineering. Comparisons between the studies are valuable, because they allow a comparison of observed correlations between desires for different attributes derived in this study with expert opinion on the extent to which these attributes can be realized in conjunction.

INTRODUCTION

For as long there have been computer programs, those developing and using them have been concerned about their quality. Despite this long-standing interest in the topic, even after fifty years of computer systems development the whole software quality area remains plagued with unanswered questions. Among the many fundamental quality issues that have not yet been properly addressed are: inconsistency in software quality factors and their definitions in software quality models [De Jong and Trauth (1993); Denning (1992); Fenton and Pfleeger (1997); Kitchenham and Pfleeger (1996)]; inconsistency in the quality models' attribute relationships; tradeoff relationships between quality attributes; and differences between the perceptions of software quality held by members of different occupational groups [Wilson and Hall (1998); Sverstiuk and Verner (2001)].

This work explores the differences in software quality perceptions between different groups of people involved with the software development process. The article compares the findings of a new survey of software stakeholders with claims made in prominent work produced by experts in the theory and practice of software quality.

BACKGROUND

Software quality is not a simple, easily measured property. To make the concept of software quality more useful and measurable, experts in the field have defined a large number of attributes associated with high quality software, such as reliability, usability and maintainability. These attributes may not be strongly associated with each other, and in practice often cannot all be realized at high levels. Existing research shows that more and more frequently we have to look at software quality not as an absolute measure, but in terms of trade-offs [Gentleman (1998)]. Ideally, we would like to have every software system possess the highest measure of quality for each software quality attribute, but in reality, everybody involved with the system, from developers to managers and users, has to compromise and focus on the most important quality factors. Even if unlimited resources were available, research suggests that some attributes are in principle impossible to maximize in the same piece of software — for example the optimization of efficiency may limit the level of reliability that can be obtained. Finding the right balance of quality attribute requirements and identifying conflicts among the desired quality attributes is an important step in developing successful software products [Boehm, In (1996)].

A number of existing studies have attempted to identify relationships between different software quality attributes. In most of these studies authors used their own experiences and expert knowledge of software quality issues to derive correlation matrixes showing relationships between software quality factors. Glass (1992); McConnell (1993); Shumskas (1992); and Perry (1991) analyzed the relationships between software quality attributes.

While our study does not attempt to determine the empirically *achieved* levels of each attribute, or even the *perceived* level of attainment in each area, it documents the extent to which a *desire* for a high level of each attribute exhibits positive or negative correlation with a desire for a high level of another attribute. This cannot in itself confirm or deny the correlations in attained levels suggested by earlier studies. But by documenting these correlations of desire, the study makes it possible to compare them with these earlier observations, and so suggest whether the overall desires of each stakeholder group are likely to be realizable. If, for example, users who attached a high priority to efficiency were also unusually likely to attach a high priority to reliability then we would have a clue as to why the development process sometimes yields software of low perceived quality.

METHOD

An online survey of 315 software stakeholders was conducted. The survey included questions covering stakeholder's job function, their relationship to software product most important for their job function, and a set of questions asking the respondent to rate the importance of each of 13 software quality attributes. Each attribute was rated independently on a scale of 1–7, where 7 meant very important and 1 meant not important.

The study addressed two research questions:

1. What correlations exist between the priorities assigned by a large sample of software stakeholders (developers, users, and managers) to different software quality attributes?

2. How do these observed correlations match with those prescribed by experts in the software quality?

The software quality attributes evaluated were:

- ACCURACY: The degree to which the software outputs are sufficiently precise to satisfy their intended use.
- TESTABILITY: The effort required to test the software to ensure that it performs its intended functions.

- USABILITY: The effort required to learn and operate this software.
- SECURITY: The extent to which access to this software by unauthorized persons can be controlled.
- EFFICIENCY: The amount of computing resources required by this software to perform its function.
- CORRECTNESS: The extent to which this software satisfies its specifications and fulfills your mission objectives.
- PORTABILITY: The effort required to transfer this software from one hardware configuration or software system environment to another.
- AUGMENTABILITY (SCALABILITY): The extent to which this software can take advantage of additional resources to deal efficiently when increased demands are placed on it.
- INTEROPERABILITY: The effort required to couple this software with another.
- ROBUSTNESS: The degree to which this software continues to function in the presence if invalid inputs or stressful environmental conditions.
- FLEXIBILITY: The effort required to modify this software for uses or environments other than those for which it was specifically designed.
- MAINTAINABILITY: The effort required to locate and fix an error in this software, or to change or add capabilities.
- REUSABILITY: The extent to which components or modules of this software can be used for other purposes.

These attributes were selected through a review of existing literature [Haigh, 2002]. Many of the attributes came from one of the most heavily cited software quality models - the Boehm et al. (1976) software quality model. Some attributes from more recent models were incorporated, and many of the descriptions were updated or simplified to make them more relevant to non-specialists and to reflect technological changes. Correlations identified between these attributes were identified through an examination of leading works in the software quality literature, and are reported below.

The survey was placed online and made available using a web interface connected to a database. The URL was distributed via email to the following groups: 1. Technical staff at the Wharton School Computing Department of the University of Pennsylvania. 2. Executive MBA students and alumni at the Wharton School of the University of Pennsylvania. The students were asked to spread the survey within their own organization. Distribution of the survey to this group of people facilitated reaching managers, users, and technical personnel from all sectors of the US economy. 3. Readers of the following internet newsgroups: comp.databases.ibm-db2, comp.databases.ms-access, comp.databases. ms-sqlserver, comp.databases.sybase, comp.human-factors, comp.software-eng, comp.software. testing, comp.software.measurement. Distribution of the survey to these newsgroups helped to reach wider population of technical personnel with experience in various application development processes.

RESULTS

This section presents the results in the following order: a summary of the background of the respondents. The review of the results continues with a discussion of the data analysis and comparison to the existing expert studies.

DEMOGRAPHIC AND RELATED DATA

Each respondent identified him- or herself as either a user or developer of the software concerned, and as either a manager (managing its users or developers) or non-manager (personally using or developing the software concerned). Combining these two variables thus divided respondents into four groups, which are referred to here as stakeholder roles: User, Manager of Users, Developer, and Manager of Development. Table 1 shows the distribution of respondents by their stakeholder roles.

Table 1. Respondent distribution by stakeholder role

Stakeholder Group	Frequency	Percent
Developer	46	14.6
Manager of Development	52	16.2
User	155	49.2
Manager of Users	59	18.7
Missing Data	3	0.9
Total	315	100

Thirty one percent of the respondents were responsible for development of the software concerned: 16.2% were managing its development, while a further 14.6% were personally performing development tasks. The remaining 69% of the respondents were not associated with the development of the software evaluated, and are therefore treated here as users. 50% personally used the software they evaluated and 18.7% identified themselves as managers of the users of the software they evaluated. (35% of the respondents fell into one or other of the management roles).

The respondents came from a variety of industries as shown in table 2.

Table 2. Respondent distribution by industry sector

Industry Sector	Frequency	Percent
IT and Telecomm	92	29.2
Government	16	5.1
Healthcare	32	10.1
Manufacturing	55	17.5
Military	5	1.6
Academic and Research	15	4.8
Service-Non-Computer	100	31.7
Total	315	100.0

Most of the respondents (60%) came from just two of the sectors: (1) IT and Telecommunications, and (2) non-IT services. Overall, however, seven major industry categories were represented.

Table 3 shows the distribution of stakeholder roles by industry. Respondents associated with developers and developer managers mainly came from IT and Telecommunication industries: 43% and 44% respectively. The service-non-computer industry was most represented for respondents not associated with software development: 39% of software users and 32% of user managers were from this industry.

Table 3. Stakeholder roles by industry

Industry (column %)	Developer (*n*=46)	Dev Manager (*n*=52)	User (*n*=155)	User Manager (*n*=59)
IT and Telecomm (*n*=92)	43.4	44.2	21.3	25.4
Government (*n*=16)	10.9	1.9	3.4	6.8
Healthcare (*n*=32)	6.5	7.7	12.3	10.2
Manufactur (*n*=55)	13.1	13.5	18.7	22
Military (*n*=5)	2.2	3.9	0.7	1.7
Academic and Research (*n*=15)	6.5	11.5	3.2	1.7
Service-Non-Computer (*n*=100)	17.4	17.3	40	32.2

DATA ANALYSIS

Bivariate correlation was used between all quality attributes, yielding a Pearson Correlation matrix. Table 4 presents results of the correlation data analysis. Significance as reported is two tailed. (+) shows a positive correlation, with significance better than the 0.05 level; (–) shows a negative correlation, with significance better than the 0.05 level; (+ +) shows a positive correlation, with significance better than 0.01 level; (– –) shows a negative correlation, with significance better than the 0.01 level; (+ + +) shows a positive correlation, with significance better than the 0.001 level; (– – –) shows a negative correlation, with significance better than the 0.001 level.

Table 4. Correlations between pairs of priorities assigned to software quality attributes (all respondents)

	Corr	**Maintn**	**Usabil**	**Testab**	**Flexibi**	**Portab**	**Reusab**	**Inter**	**Integ**	**Accur**	**Robust**	**Augme**	**Effic**
Correctness	🟦	– –	+ +	– – –	– – –	– – –	– – –	+	– –	+ +	+ +	– – –	
Maintainability	– –	🟦	– – –	+ +		–		– – –		– – –	– – –		
Usability	+ +	– – –	🟦	– – –	– – –		– – –	+ +		+ +	+ +	– –	– –
Testability	– –	+ +	– – –	🟦	+		+	– – –		– – –	– – –		
Flexibilty	– –		– – –	+	🟦		+ +	– – –	– – –	– – –	– – –		
Portability	– –	–				🟦				– –		–	– –
Reusability	– –		– – –	+	+ +		🟦	– –	– – –	– – –	– – –	+	
Interoperability	+	– – –	+	– – –	– –		– –	🟦				– – –	– –
Integrity	– –		+ +		– – –		– – –		🟦	– –	–	+	
Accuracy	+ +	– – –	+ +	– – –	– – –	– –	– –		– –	🟦	+ +	– – –	
Robustness	+ +	– – –	+ +	– – –	– – –		– – –		–	+ +	🟦	– –	
Augmentability	– –		– –			–	+	– – –	+	– – –	– –	🟦	+ +
Efficiency			– – –			– –		– – –				+ + +	🟦

DISCUSSION

Table 4. shows that maintainability and testability exhibit strong positive correlation. Likewise, correctness and accuracy proved to be positively correlated with each other, but negatively correlated with both testability and maintainability. Correctness and accuracy were also positively correlated with usability – another attribute favored by users and those with less involvement or interest in software quality issues. While robustness showed no significant variation according to any of the independent variables, it too is positively correlated with correctness, accuracy and usability and negatively correlated with maintainability and testability. What these four attributes have in common is an obvious association of the words involved with everyday notions of quality. They were negatively correlated with flexibility, portability, reusability and augmentability – all factors less likely to appear important to those without any understanding of the software development process.

The most striking result is therefore the very close relationships observed within two attribute groups: the first one consisting of maintainability and testability and the second one consisting of correctness, accuracy, robustness and usability. The members of each of these groups show a strong positive correlation with each other and are also very similar in their correlations with each of the attributes outside the group. Interoperability is very strongly correlated with usability and less strongly correlated with correctness – putting it close to the correctness/accuracy/usability group. Flexibility and reusability form a third group, negatively correlated with most of the other attributes. Both of these attributes were consistently rated among the least important – this may explain the general negative correlation but does not in itself explain why they correlate positively with each other.

COMPARISON OF OBSERVED ATTRIBUTE PRIORITY CORRELATIONS WITH EXISTING LITERATURE

A number of existing studies have attempted to identify relationships between different software quality attributes. In most of these studies authors used their own experiences and expert knowledge of software quality issues to derive correlation matrixes showing relationships between software quality factors. Glass (1992); McConnell (1993); Shumskas (1992); and Perry (1991) analyzed the relationships between software quality attributes. Table 5 presents summary of the comparison of the existing studies with this research.

Table 5 shows that experts agree on many correlations, while contradict each other on others. For example, both Shumskas and Perry suggested that maintainability was positively correlated with testability – a relationship strongly echoed in the quality priorities reported by the respondents in the present study. Likewise, both these authors suggested that integrity was negatively correlated with flexibility, another finding echoed by respondents. McConnell suggested a positive correlation between attained levels of correctness and accuracy, and Perry a relationship between correctness, robustness and usability. All three of these claims were reflected in the quality priorities reported in the results of our study. Respondents of this study reported negative relationships between efficiency and interoperability (in accordance with Perry and Shumskas), usability (in accordance with Perry and Glass), portability (in accordance with Perry and Glass) and correctness (in accordance with McConnell).

Table 5. Comparoson of expert options with each other and with this stuly

	Class	**mcConnel**	**Shumskas**	**Perry**	**This study**
Accuracy (positive correlation)		+ Correctness			+ Correctness, Usability, Robustness
Accuracy (negative correlation)		– Efficiency, Robustness			– Flexibility, Maintainability, Reusability, Testability, Portability, Integrity
Correctness (positive correlation)		+ Accuracy	+Integrity Testability, Flexibility, Reusability	+ Robustness, Usability, Maintainability, Testability, Flexibility	+ Accuracy, Robustness, Usability, Interoperability
Correctness (negative correlation)		– Robustness, Efficiency			– Flexibility, Portability, Reusability, Testability, Maintainability, Integrity, Efficiency
Efficiency	– Portability, Robustness, Usability, Testability	– Correctness, Integrity, Accuracy, Robustness	– Maintainability, Testability, Interoperability, Portability	– Integrity, Usability, Maintainability, Testability, Flexibility, Portability, Reusability, Interoperability	–Interoperability, Usability, Portability, Correctness
Flexibility (positive correlation)			+ Correctness, Maintainability, Interoperability	+ Correctness, Robustness, Usability, Testability, Reusability	+ Testability, Reusability
Flexibility (negative correlation)			– Efficiency, Integrity	– Efficiency, Integrity	– Correctness, Usability, Accuracy, Robustness, Integrity, Interoperability
Integrity (positive correlation)			+ Robustness, Correctness		
Integrity (negative correlation)		– Efficiency	– Efficiency, Flexibility, Interoperability, Reusability	– Efficiency, Flexibility, Reusability, Interoperability	– Reusability, Flexibility, Correctness, Accuracy, Robustness
Interoperability (positive correlation)			+ Flexibility, Interoperability	+ Portability	+ Usability, Correctness
Interoperability (negative correlation)			– Efficiency, Integrity, Robustness	– Efficiency, Integrity	– Maintainability, Testability, Reusability, Flexibility, Efficiency

	Class	mcConnel	Shumskas	Perry	This study
Maintainability (positive correlation)			+ Usability, Correctness, Testability, Flexibility, Reusability	+ Correctness, Robustness, Usability, Testability, Flexibility, Portability, Reusability	+ Testability
Maintainability (negative correlation)			– Efficiency	– Efficiency	– Correctness, Accuracy, Interoperability, Robustness, Usability, Portability
Portability (positive correlation)			+ Reusability	+ Maintainability, Testability, Reusability, Interoperability	+ Interoperability
Portability (negative correlation)	– Efficiency		– Efficiency	– Efficiency	– Correctness, Maintainability, Accuracy, Efficiency
Reusability (positive correlation)			+ Robustness, Correctness, Maintainability, Interoperability, Portability	+Maintainability, Testability, Flexibility, Portability	+ Flexibility, Testability,
Reusability (negative correlation)			– Efficiency, Integrity	– Robustness, Efficiency, Integrity	– Correctness, Usability, Accuracy, Robustness, Integrity, Interoperability
Robustness (positive correlation)		+ Usability	+ Maintainability, Reusability, Integrity	+ Correctness, Ustability, Maintainability, Testability, Flexibility	+ Correctness, Accuracy, Ustability
Robustness (negative correlation)	– Efficiency	– Accuracy, Correctness, Efficiency, Integrity	– Interoperability, Efficiency, Testability, Flexibility	– Reusability	– Flexibility, Maintainability, Reusability, Testability, Integrity
Testability (positive correlation)			+ Maintainability, Reusability	+ Correctness, Robustness, Ustability, Main-tainability	+ Reusability, Maintainability, Flexibility
Testability (negative correlation)	– Efficiency, Usability		– Efficiency, Robustness	– Efficiency	– Correctness, Ustability, Accuracy, Interoperability, Robustness
Usability (positive correlation)		+ Accuracy	+ Maintainability, Testability, Flexibility	+ Correctness, Robustness, Maintainability, Testability, Flexibility	+ Correctness, Accuracy, Interoperability, Robustness
Usability (negative correlation)	– Efficiency, Testability		– Efficiency	– Efficiency	– Maintainability, Flexibility, Reusability, Testability

One main set of exceptions was noted. As reported above, respondents showed strong negative correlations between the two groups of accuracy/

correctness/robustness/ usability and maintainability/testability. While the positive relationships between maintainability and testability were supported by the previous studies, as were positive relationships between correctness and accuracy, other aspects of these findings were less supportable. McConnell suggests that those of attained levels of accuracy and correctness are negatively correlated with robustness. Similarly, Perry believes that correctness and usability are positively correlated with maintainability and testability (supported with respect to the latter by Shumskas). The respondents in the present study, however, show negative correlations between the priorities assigned to these attributes. The views of the experts here seem to make sense.

Further analysis of survey results (not reported here for reasons of space) suggested that accuracy/correctness/robustness/usability was favored by less experienced respondents and end users, while maintainability/testability was favored by more experienced respondents and development managers. Within the samples of developers and development managers the results were more in keeping with those suggested by the experts. As we saw, earlier studies were based on the experience of their authors as developers and observers of development projects rather than a sampling of the views of any broader population, and so we should not be surprised that the views of the experts were closer to those of development staff (whom they more closely resemble) than those of users.

CONCLUSION

This work explores the differences in software quality perceptions between different groups of people involved with the software development process. Three hundred and fifteen respondents ranked each of thirteen generally accepted attributes of software quality on a scale of one to seven according to their perceived importance for the piece of software most vital to that individual's work. The results of this study were compared to the results of the existing expert studies.

Comparisons between these studies and the present research must be made with caution. The present study can neither test nor confirm these earlier models because it examines the quality attributes most prized by different respondents, rather than those that they believe to have been obtained or to be obtainable. Despite this, comparisons between the studies remain valuable, because they allow a comparison of observed correlations between desires for different attributes derived in this study with expert opinion on the extent to which these attributes can be realized in conjunction.

Our comparison of the existing expert studies and our research revealed two main findings. Firstly, the various experts reviewed here differed considerably on the extent to which the attainment of one software quality attribute was likely to assist or hinder the attainment of another. For example, while Perry believed testability to be positively coordinated with robustness, Shumskas claimed that the relationship was negative. Second, the correlations (positive and negative) observed in this study between the priorities attached to different attributes rarely conflict with the relationships in attainable quality levels set out by the expert in earlier research. While the present study found many correlations between attributes not correlated in the other studies, there were relatively few instances in which a negative correlation in this study was accompanied by a positive correlation in the other studies, or vice versa. This suggests that the concepts of software quality held by software stakeholders are not inherently unrealizable, in as much as correlations between desire for specific software quality attributes were broadly in line with expert opinion on natural correlations between attainable quality levels.

REFERENCES

1. *Arthur I.J.* Measuring Programmer Quality. — New York: John Wiley and Sons, 1985.
2. *Boehm B.W., Brown J.R. and Lipow M.* (Eds.). Quantitative Evaluation of Software Quality. Proceedings of the Second International Conference of Software Engineering. — 1976. — P. 592–605.
3. *Boehm B.W., In H.* Identifying Quality-Requirement Conflicts // IEEE Software. — 1996. — **13**, № 2. — P. 25–35.
4. *De Jong K. and Trauth S.L.* Culture Shock: Improving Software Quality // The Journal of the Quality Assurance Institute, April 1993. — 7(2). — P. 24–30.
5. *Dekkers N.* Maximising Customer Satisfaction // Proceedings of the 12th European Software Control and Metrics Conference in London Eds K. Maxwell, S. Oligny, R. Kusters and E. van Veenendaal, April 2-4th, 2001.
6. *Denning P.J.* What is Software Quality // Communications of the ACM. — 1992. — **35**, № 1. — P. 13–15.
7. *Deutsch M.S and Willis R.R.* Software Quality Engineering. — New York: Prentice-Hall Englewood Cliffs. — 1988.
8. *Fenton N.E. and Pfleeger S.L.* Software Metrics // A Rigorous and Practical Approach, 2nd Edition., New York: PWS Publishing Co. — 1997.
9. *Gentleman W.M.* If software quality is a perception, how do we measure it? // The Quality of Numerical Software: Assessment and Enhancement, Ronald Boisvert, ed., Proceedings of IFIP WG2.5 Working Conference 7, Oxford, UK, 7–12 July 1996, Chapman & Hall, London. — P. 32.
10. *Glass R.L.* Building Quality Software — N.Y.: Prentice Hall, 1992.
11. *Haigh M.* Software Quality Revisited: Diverging Priorities Between Stakeholder Groups? // Ph.D. Dissertation, Drexel University, 2002.
12. *Jacobs S.* Introducing Measurable Quality Requirements: A Case Study // IEEE International Symposium on Requirements Engineering. — 1999. — June 7–11, Limerick, Ireland.
13. *Kitchenham B., Pfleeger S.L.* Software Quality: The Elusive Target // IEEE Software, January 1996. — P. 12–21.
14. *Kusters R.J., van Solingen R., Trienekens J.J.M.* User-perceptions of embedded software quality // Eighth IEEE International Workshop on Software Technology and Engineering Practice incorporating Computer Aided Software Engineering. — 1997. — P. 184–97.
15. *McConnell Steve.* Code Complete: A Practical Handbook of Software Construction. — Redmond. — WA: Microsoft Press, 1993.
16. *Perry W.E.* Quality Assurance for Information Systems: Methods, Tools and Techniques // QED Technical Publishing Group, 1991.
17. *Shumskas A.F.* Software Risk Mitigation // Schulmeyer, G. Gordon and James I. McManus, ed. Total Quality Management for Software. — NY: Van Nostrand Reinhold. — 1992. — P. 190–220.
18. *Sverstiuk M., Verner J.* Modelling Software Quality Through Organizational Position and Software Role: A Pilot Study // 12th European Software Control and Metrics conference. — London. England, April 2001.
19. *Wallmuller E.* Software Quality Assurance: A Practical Approach // Prentice-Hall Englewood Cliffs NJ 1994.
20. *Wilson D.N. and Hall Tracy.* Perceptions of Software Quality: A Pilot Study // Software Quality Journal. — 1998. — № 7. — P. 67–75.

Received 11.07.2007

From the Editorial Board: the article corresponds completely to submitted manuscript.