Modelling Software Quality Through Organisational Position and Software Role: A Pilot Study

Maria Sverstiuk, June Verner

Abstract

Over the last 30 years, many software quality models have been defined. Because there has been little empirical validation of these models, we are left with a number of unanswered questions. How do managers, developers, and users really perceive software quality? Is there an overarching view of quality that colours their view of software quality attributes? Does their role with a particular software product make a difference to their view of software quality? Does their major job function also colour their attitude? In short, we want to better understand what is meant by the term "software quality" and to analyse perceptions of what constitutes software quality from the perspective of different groups of professionals involved with both software development and software use. In order to investigate these aspects of quality we conducted a survey of 300 professionals asking them to rank the importance of 17 software quality attributes.

The results of our study show different patterns for our respondent groupings. Although our respondents may discuss the same quality attributes, they appear to have different views of the importance of these attributes. While some of the software quality attributes appear to be important to all professionals regardless of their job function or software role, there are differences in software quality priorities for different professional and user groupings. Analysis of our results shows that the overarching view of software quality taken by most of our respondents is a "user view".

1. Introduction

Quality is defined as "a distinctive inherent feature"; as "a degree of excellence"; as well as "a degree of conformance to a standard (as of a product or workmanship)" [17]. Being "a degree" of something implies the relative nature of the concept. The multifaceted nature of "quality" is behind a number of unresolved issues that directly impact the quality of software products. A review of the existing research shows that it is common to view software quality not as an absolute measure, but in terms of trade-offs [5]. Ideally, we would like every software system to possess the highest measure of quality for each software attribute. However, in reality, everybody involved with a software system, from developers to managers and users, has to compromise and focus on the quality factors most important to them. Because managers, technical personnel and customers differ in their views of what constitutes a quality product the differences in their perceptions and the impact of these perceptions on software product development require greater understanding. The relative nature of quality suggests that software quality needs be defined from many points of view, depending on the role the person plays in the development process and on the type of system being developed ([1], [4], [16]). Garvin, [5] noted differences in quality perceptions in a quality framework where he identified five main overarching views of quality: (1) transcendental, (2) product, (3) user, (4) manufacturing, and (5) value-based views.

The main objective of our research is to analyse perceptions of the constitution of software quality from different points of view. Our study is a step in the direction of achieving a more

complete picture of software quality and a broader perspective of software quality using statistically validated results. A better understanding of software quality priorities will lead to better communication between the different parties involved with the system. Managers and developers should understand what aspects of software quality are important to them, and to users, so that they can ensure that developers of the system implement the most important features.

2. Related Research

A number of studies have analysed software quality factors and presented tradeoffs in matrix form (e.g., [8], [11], [13]). Glass [8], basing his work on Boehm et al's [2] quality model, analyzed the relationships between software quality product factors and provided a chart representing tradeoffs in achieving desired quality attributes. Glass [8] also prioritised software quality attributes within the software development lifecycle stages and provided a correlation matrix showing those quality attributes that reinforce each other, and those attributes that have no effect on the other quality attributes. McConnell [11] also looked at the relationships between Boehm et al's [2] software quality attributes and, based on experience, derived a quality factor correlation matrix. Shumskas [14] also presented a version of Boehm et al's [2] software quality attributes and included a matrix showing not only positive and negative attribute correlations, but also those quality relationships that are application dependent. Perry [13] (based on his intuition), summarized relationships between the quality factors in Boehm et al's [2] model as inverse, neutral, and direct. He notes that there are lifecycle aspects of software quality factors that we must consider, where the quality factor should be measured, and where the impact of poor quality is realised.

In a later section, we extend our discussion and compare our results with the earlier research.

2.1. Unanswered Questions

Because there has been so little empirical validation of the suggested software quality models [15] our literature review has left us with a number of unanswered questions. How do we think about software quality? What are different groups' mental models of software quality? Are there differences in the groups' overarching views of software quality? Do the software quality attributes correlate with each other as individual authors' experiences suggest? If the mental models of different groups of people truly are different, we need to know so that we can understand how best to communicate unambiguously. How can we possibly meet software quality requirements if we have little understanding of the mental models held by the people who manage, develop and use software?

Our pilot survey was conducted in order to gain insight into software users, software development practitioners' and IT managers' cognitive models of software quality. This will allow us to explore software quality in order to draw conclusions based on the experiences of a number of different groups of people.

In the next section we focus on the method of study and follow this with a discussion of our experimental results and our conclusions.

3. Research Method

We developed a survey that includes questions covering our respondent's overarching software quality views, job function, their relationship to a specific software product that they

recently used, and a set of questions asking the respondent to rate the 17 software quality factors. We distributed the survey via email to the employees of the University of Pennsylvania Wharton School Computing Department, to the graduate and undergraduate students of the Drexel College of Information Science and Technology, and to recent graduates of the Wharton Executive MBA program. Such a distribution guaranteed us respondents with different technical backgrounds.

Our questionnaire was mainly based on the Boehm et al. [2] software quality model because, even today, it is still one of the most heavily cited quality models in the literature. We asked the respondents to rank, on a seven point likert scale, the importance of seventeen software quality attributes from the perspective of their job, with reference to a software system with which they are familiar. A score of one is given to an attribute that has no relevance to the respondent, and a score of seven to an attribute with great importance. Definitions of each quality attribute, based on those given in the original Boehm et al model [2], were included in our questionnaire. Though we did not make any changes to the original definitions, we do wonder if some definitions should be updated due to the changes in technology since the mid-seventies. In a separate question, we asked respondents to identify their role in relation to the software they are assessing, that is, manager, user, or developer. We also presented respondents with the list of Garvin's [5] definitions of quality and asked them to identify which view most closely coincides with their own view of "software" quality.

4. Results

The survey return rate was 56%, with 305 responses. Our respondents had backgrounds that varied from corporate and technical management, and software development, to casual users of software. The respondents were employed in a number of different industries: manufacturing (9%), banking (10%), education (25%) and computer-related areas (37%). Table 1 shows the distribution of our respondent's software roles. We were interested in three "software roles" with respect to the evaluated software: manager, user, and developer. Though 22% of our respondents occupy managerial positions (see Table 2), in fact, only 10% had managerial responsibilities for the software they evaluated. Seventy-seven percent of respondents identified themselves as users of the software they were assessing; 6% of the respondents were developers of the software. This gave us a group of respondents whose roles spanned all main categories: manager, developer and user and allowed us to compare their perceptions.

Table 1. Respondents' roles with respect to the evaluated software

	Frequency	**Percent**
Manager	29	10
User	234	77
Developer	17	6
Total	280	92

We identified four "job functions": 1. corporate/technical manager, 2. data administrator/application developer, 3. IS/MIS/DP/IT staff, and 4. other non-technical personnel. Table 2. shows the distribution of the respondents' job functions. A particular job function might not always coincide with the respondent's responsibility concerning the software system that the respondent focussed on, in the questionnaire responses.

Table 2. Respondents' job function

Job Function	Frequency	Percent
Corporate / Technical Manager	67	22
Data Administrator / application developer	19	6
IS/MIS/DP/IT Staff	72	24
Other non-technical	147	48
Total	305	100

Table 3. shows the distribution of "software role" within each "job function". As mentioned earlier though 22% of our respondents occupy managerial positions, only 10% had managerial responsibilities for the software they evaluated. The majority of respondents identified themselves as users of the software that they assessed. Twenty four percent of our respondents were both managers by function and by their software role; 15% of the IS staff were managers of the software they assessed and 4% of them participated in the development of this software.

Table 3. Distribution of the user roles by job function

Job Function \ User Role	Manager	User	Developer
Corporate / Technical Manager	24%	70%	6%
Data administrator / application developer		74	26%
IS/MIS/DP/IT Staff	15%	81%	4%
Other	2%	94%	4%

As the main goal of this study is to explore the differences in the software quality attribute priorities within our different professional groups we divided our sample into groups in three different ways:

- by the respondent's role with regard to the evaluated software (either developer, user, and manager, see Table 1; and
- by job function (i.e., corporate/technical managers, data administrator/application developer, IS/MIS/DP/IT staff and non-technical (other), see Table 2; and,
- by Garvin's [5] overarching quality view (transcendental, product, user, manufacturing and value).

With these groupings, we can identify how the ratings of software quality attributes changes with differences in roles and job function.

Figure 1 shows means for all 17-quality attribute rankings for the three software role groups. For this data set, there are three significant differences (< 0.5) in responses from the groups. These are shown with a '**' on the X-axis of Figure 1. This figure shows that there are many similarities in the groups' perceptions: for example, correctness and accuracy were consistently ranked as most important across all three role groups. Robustness was next most important for managers, and developers, while usability came next for users. Respondent perceptions of adaptability and accessibility were also very similar across software roles. However, portability though scored as important by developers and users was seen as less important by managers. On the other hand, for developers communicativeness was much less important than for managers and users. Differences were also observed for testability and interoperability; these attributes were more important for managers and developers than for users.

The attributes ranked highest by users include usability, adaptability reusability and accuracy. Developers ranked accessibility, efficiency, interoperability, maintainability, and testability higher than the other groups while managers ranked all the other attributes highest.

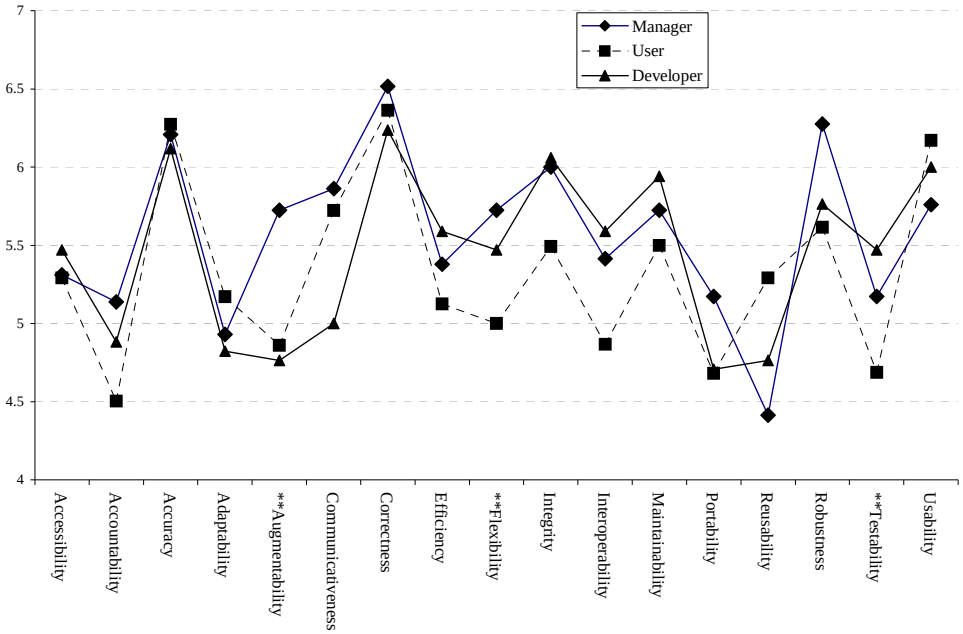

Figure 1. Quality Attribute Means by Software Role

Figure 2, displays overall similarities and differences in perceptions of the software quality attributes, by job function, and shows that actual job function makes quite a difference in the ratings when compared with our earlier software role. Corporate/technical managers exhibit the most noticeably different quality preferences when compared with other professional groups. As Figure 2 shows they rated almost all the quality attributes lower in importance than the other groups except for accuracy, communicativeness, correctness, and robustness; both of which were rated similarly to the other groups. There are many more significant differences in responses for these job function groups (12 out of 17). Again significant differences < 0.05 shown with '**'. As Figure 2 shows, respondents whose job function has to do with software in either an IS/MIS/DP/IS staff role or as data administrator/application developers, rate quality attributes higher than the other two respondent groups. The mean ratings here are quite a different from the mean ratings shown Figure 1. Figures 1 and 2 together show that both the software role taken by the respondent with regard to software, and the respondent's job function impact differently on quality attribute percepetions. However, for a comprehensive study of how the overlaps between job function and software role affect attitude differences more data would be required.

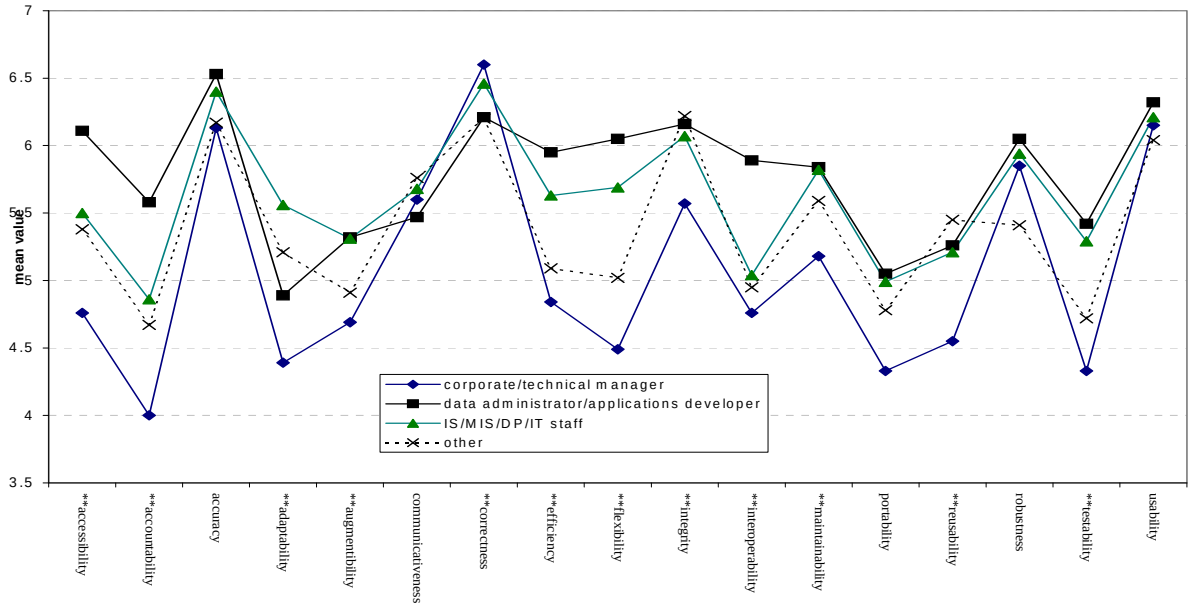

Figure 2. Quality Attribute Means by Job Function

When we consider the overarching view of software quality taken by our respondents we note that 36% of our respondents were not comfortable taking a single view of quality; 29% preferred to combine two views of quality, 5% three views and 2% four or five views. Figure 3 shows our results for quality view by software role and job function.

The majority of our respondents, 58%, have a **user view** of software quality, i.e., that it is a measure of how well software product characteristics meet the users' needs. Eighty one percent of corporate/technical managers, 100% of data administrator/application developers, 81% of IS/MIS/DP/IS staff and 84% of "other" take this view of quality. When categorized by software role, 69% of managers, 86% of users and 82% of developers take a user view of software quality.

The **trancendental view** that software quality can be recognised but not defined, taken by only 4% overall of our respondents is made up of 4% of the corporate/technical managers, 8% of the IS/MIS/DP/IT staff, and 7% of "other". When categorized by software role, a transcedental view is taken by 8% of managers, 6% of users, and 6% of developers.

The **product view**, that software quality is a context-independent measure of software product internal and external properties, is taken by 16% corporate/technical managers, 11% of data administrator/application developers, 18% IS/MIS/DP/IS staff and developers take a product view of software quality 18% "other". When categorized by software role 38% managers, 14% users and 18% developers take a product view.

The **manufacturing view**, that software quality is a measure of how cost efficient the product development process was, is taken by 10% of our respondents. A manufacturing view is taken by 8% of the corporate/technical managers, 21% of the data administrator/application developers, 16% of the IS/MIS/DP/IS staff and 16% of "other". When categorized by software role 14% managers, 14% of users and 24% developers take a manufacturing view of software quality.

The **value view**, that software quality is a trade off between the cost of software development and how much a customer values certain software features, is taken by 16% of our respondents. A value view of software quality is taken by 27% of the corporate/technical

managers, 16% of the IS/MIS/DP/IS staff, 14% of the data administrator/application developers, and 28% of "other". When categorized by software role 24% of managers, 23% of users and 29% of developers take a value view of software quality.

These results show that a user view of quality i.e., how well the software product meets users needs is the most prevalent view taken of software quality (58%). The other views of quality are less common; the user view is followed by the value view at only 16%, the product view at 12%, the manufacturing view at 10%, and the transcendental view at 4%.

When we consider software roles 69% of managers, 86% of users and 82% of developers take a user view of quality. The next most common view taken by this categorization of respodents viewa are: managers-product view 38%, users-value view 23%, and developers-value view 29%.

When we consider job function 81% of corporate/technical managers, 100% of data administrator/application developers, 81% of IS/MIS/DP/IS staff and 84% of "other" take the user view of quality. The value view is next most popular, with 27% corporate/technical managers, and 28% of "other" choosing this view The IS/MIS/DP/IS staff are alittle different with 18% taking a product view. Figure 3 below shows both job function and software roles by overarching software quality views. This figure shows very clearly that the user view is the main software quality view.

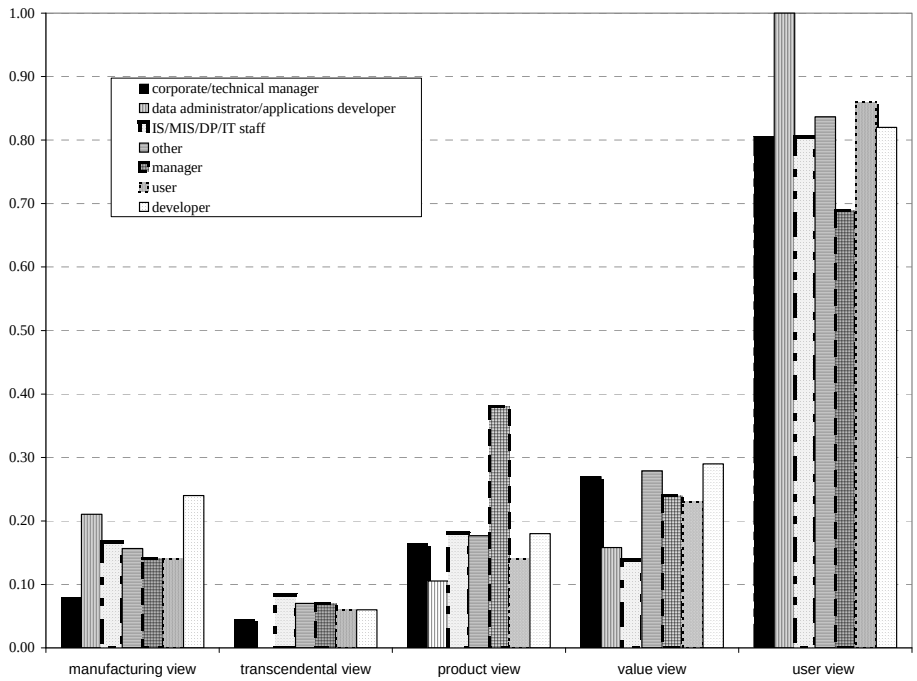

Figure 3. Quality View by Software Role and Job Function

We next compare the 17 software quality attributes within the five overarching quality views see Figure 4, below.

We find that respondents who take a transcendental view of software quality, i.e., that "software quality can be recognised but not defined" rate almost all the quality attributes lower than do respondents who take other views. This groups' perception of software quality will most likely make it difficult for software developers to deal with them. If this group cannot define quality, have to see the product first to know if quality is there, and rate the majority of quality attributes lower than other groups, software development becomes problematic. This is an interesting result but needs further research with additional data. In

contrast respondents who take a manufacturing point of view tend to rate the quality attributes highest of all. Not only do they want to measure how cost efficient the development process is, they also want a high level of quality for their money. We may assume that these respondents are interested in development efficiency/quality tradeoffs. The value and user views give very similar ratings for the software quality attributes.

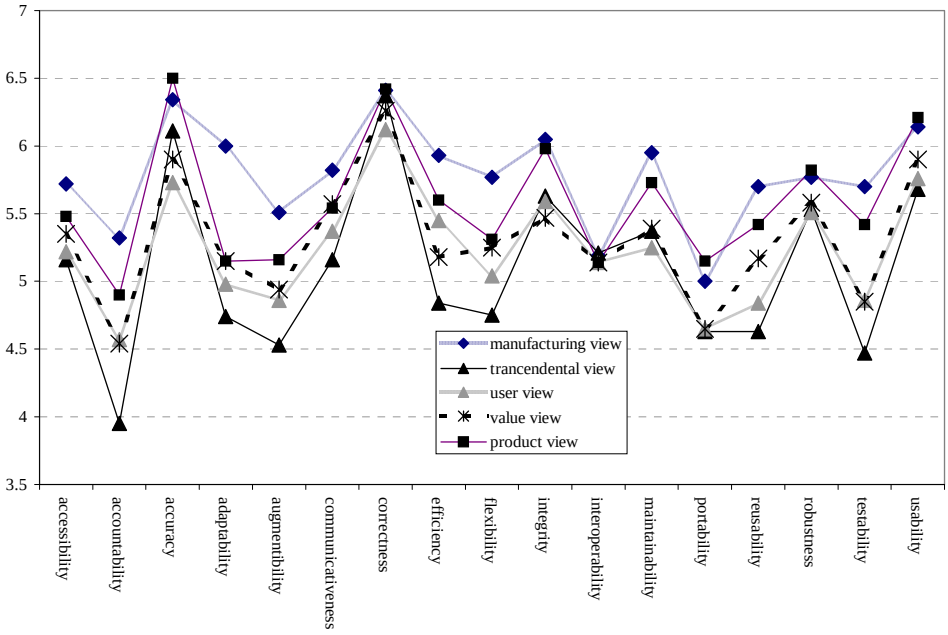

Figure 4. Software Quality Attributes by Overarching Quality View

While the main purpose of our study is to explore our respondents' prioritisation of software quality attributes, we also compare the results of our derived software quality attribute relationships with the results of earlier studies. We show a matrix, similar to that of the earlier research, see Table 4. A "**" in our matrix symbolises correlations at the 0.01 level (two tailed). The authors of the prior research do not completely agree with each other on all interrelationships between software quality attributes ([7], [8], [11], [13], [14]). For instance, Perry [13] disagrees with Gillies [7] on the correlation between Portability and Testability; our results are consistent with Perry [13] as we found a positive correlation between *portability* and *testability*. Though overall, our results are mostly agree with Perry's [13] findings there is one noticeable difference. When we analyse the relationship of *efficiency* with the other attributes, our results did not match any of the earlier work. Although all authors report that efficiency has a negative effect on all other quality factors our results did not support this as efficiency was positively correlated with all software quality attributes. In fact, we did not get any negative correlations at all.

Table 4. Software quality attributes correlations

	Efficiency	Reliability	Testability	Portability
Correctness	0.105	0.169**	0.089	-0.018
Efficiency	1	0.185**	0.422**	0.05
Integrity	0.355**	0.173**	0.322**	0.132**
Usability	0.9	0.087	0.148**	0.211**
Maintainability	0.307**	0.156**	0.572**	0.337**
Testability	0.422**	0.232**	1	0.406**
Flexibility	0.421**	0.106	0.563**	0.402**
Portability	0.05	0.073	0.406**	1
Reusability	0.253**	0.163**	0.308**	0.409**
Interoperability	0.27**	0.155**	0.35**	0.371**
Adaptability	0.383**	0.05	0.458**	0.351**
Accuracy	0.189**	0.431**	0.247**	0.073
Reliability	0.185**	1	0.232**	0.073
Accessibility	0.382**	0.361**	0.394**	0.194**
Accountability	0.439**	0.227**	0.436**	0.225**
Augmentability	0.304**	0.374**	0.513**	0.366**
Communicativeness	0.149**	0.285**	0.291**	0.217**

4. Conclusions

No matter how we categorise our respondents by job function, software role, or by overarching quality view, accuracy and correctness are the software quality factors rated as most important. Usability follows for most groups although our software role group "managers" rates communicability as more important than usability. If we consider our results in more detail, we find that when we compare our respondents by (1) job function with (2) software role, we see a number of differences in our results. When job function is the grouping variable, we see many more significant differences in quality attribute rankings. This indicates that what the role of the respondent is, in the organisation, makes a greater difference to the importance placed on software quality attributes, than does their own personal interactions with software (software role).

The overarching view of quality taken by most professionals, no matter what their job function is that of "user view" i.e., concern with how well software product characteristics meet the users' needs. Respondents who take a transcendental view of quality, and rate the majority of software quality attributes lower than the other "view" groups, will probably make it difficult for software developers to provide them with (what in their view is) a quality product. This group cannot define quality, and have to see the product first to know if quality is there. This makes it problematic for our data administration/software development group given that not a single respondent in that group viewed software quality as transcendental. This is an interesting result but needs further research with additional data.

When we consider quality attribute tradeoffs we find that our results agree with those of Perry [13] except for the direction of the efficiency relationships. Given that this is a pilot study this result needs further investigation with both more data and with a different set of questions. How much the respondents'view of quality is impacted by rapid changes in technology and how the definitions of quality factors in the older models might need to be changed should be further investigated

Our study has perhaps shed some light on attitudes to software quality but there is much more work that needs to be done. To discover more about software quality we need to investigate the structure of software quality attributes within Garvin's overarching quality

framework. This will allow us to find out, for the various groups involved with software and its development, what differences an overarching quality view makes to their cognitive mappings of the software quality attributes.

References

[1] Arthur, L. J. Measuring Programmer Quality, John Wiley and Sons New York 1985

[2] Boehm, B.W., Brown, J.R., and Lipow, M. (Eds.), (1976), “Quantitative Evaluation of Software Quality”, Proceedings of the Second International Conference of Software Engineering, pp. 592-605.

[3] Cerpa N. and Verner J.M., “Practitioners’ Views of Software Quality”, The International Journal of Computing and Engineering Management, Vol. 4, No. 3, September-December 1996, pp. 1-15.

[4] Deutsch, M. S and Willis, R. R. Software Quality Engineering Prentice-Hall Englewood Cliffs NJ (1988).

[5] Garvin, D. “What Does “Product Quality” Really Mean?” Sloan Management Review, Fall 1984, pp25-45.

[6] 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.

[7] Gillies, A. Software Quality Theory and Management, 2nd Ed., London: International Thomson Computer Press, 1997.

[8] Glass, R.L. “Building Quality Software”, Prentice Hall, 1992

[9] Kitchenham B., Pfleeger S.L., “Software Quality: The Elusive Target”., IEEE Software, January 1996, pp.12-21.

[10] 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, p.184-97, 1997

[11] McConnell, Steve. Code Complete: A Practical Handbook of Software Construction. Redmond, WA: Microsoft Press, 1993.

[12] Nance, K.L., Strohmaier, M., " End user perception and software quality assessment", Journal of International Information Management vol.6, no.1, p.1-8, 1997.

[13] Perry W.E. “Quality Assurance for Information Systems: Methods, Tools and Techniques”, QED Technical Publishing Group, 1991.

[14] Shumskas, A. F. “Software Risk Mitigation” in Schulmeyer, G. Gordon and James I. McManus, ed. Total Quality Management for Software. 1992: NY, Van Nostrand Reinhold. pp. 190-220.

[15] Verner, J.M., Moores, T.T. and Barrett, R., “Software Quality: Perceptions and Practices in Hong Kong”, in Achieving Quality in Software (eds) S. Bologna and G. Bucci Chapman and Hall 1996, pp. 77-88.

[16] Wallmuller, E., Software Quality Assurance: A Practical Approach, Prentice-Hall Englewood Cliffs NJ 1994.

[17] Webster's Third New International Dictionary, Unabridged, Copyright 1993 Merriam-Webster, Incorporated.

[18] Wilson, D. N. and Hall, Tracy, "Perceptions of Software Quality: A Pilot Study". Software Quality Journal, 7 pp. 67-75, 1998