Measuring Perceived Software Quality

M Xenos and D Christodoulakis
Department of Computer Engineering and Informatics, University of Patras, Rion 26500, Greece
e-mail: xenos@cti.gr

This paper presents a method for measuring customer's perception of software quality. We argue that, although the importance of the perceived product quality is recognised world–wide, there does not exist a rigorous method for measuring customer perception of product quality. This paper presents a method expanded to measure not only end–users perception for the product, but also company employees perception for the quality of the internal deliverables produced within the company. Additionally, in this paper we present examples of the method's application on a set of projects, in parallel with internal measurements, using a set of commonly used product metrics. Subsequently, we compare measurement results that are derived from customer perception measurements to results that are derived from internal measurements, and we discuss the advantages and disadvantages of each method.

Keywords: software quality assurance, software metrics, user satisfaction measurements

Many definitions of software quality have been published, which in general agree on what quality means and their agreement can be enshrined by the phrase *'satisfaction of customer requirements'*. In simple language, software must do what the customer expects it to do. The customer plays an important role in software quality. The international standards ISO9000[1,2], IEEE[3] and Baldrige[4,5] place emphasis on customer perceived quality and expect that customer satisfaction be strongly linked to all functions of a business. Within the scope of a company's quality assurance program however, customers are not only the end–users of the product, but also the employees that use the results available at the end of each phase of the software life cycle. Therefore, implementation teams are the customers that use deliverables produced by the design team and, in turn, implementation teams produce deliverables for their customers which are the testing and maintenance teams. Throughout this paper the term 'customer' will be used in such a broad manner, including the internal company teams acting as customers as well as the end–users of the product.

This producer–customer relationship requires a method in order to measure the customer perception of quality. The ability to initially measure and eventually control customer perceived quality, is a major success factor in software business. Despite the indications derived from internal measurement, the end–user is the ultimate judge of product quality. In cases of disagreement between internal measurement and end–user perceived quality, the best company choice is to conform with end–user opinion. Furthermore, modern software companies need to measure not only the perception of product quality by end–users, but also the perception of company employees for the quality of internal deliverables. As previously explained, such employees operate as customers who receive the deliverables which other employees produce.

Quality assurance teams in modern software companies measure product quality by applying a method which relates internal measurable quantities with external quality characteristics. Many examples of such metrics and their interpretation can be found in software measurements literature. For example, function points[6] are used in order to estimate product cost, cyclomatic complexity[7] is used in order to estimate software complexity and maintainability, Halstead[8] Effort Estimator is used in order to identify required effort and time, etc. The great majority of quality assurance teams follow a standard methodology that guides them on how to organise and perform measurements and on how to relate measurement results to product quality characteristics aiming to control. Although they recognise the importance of measuring customer perceived quality, surveys measuring it are, however, not performed with a similar rigorous approach. Furthermore, companies rarely measure the quality of internal deliverables by handling them as products and employees receiving them, as customers.

Therefore, a method is needed allowing software companies to rigorously measure the customer perception of product quality. Such measurements could also be used for internal deliverables quality assessment, as well as for evaluating the performance of measuring procedures and calibrating internal product metrics used by the quality assurance team. This paper presents a method aiding in rigorous organisation of customer perceived quality measurements. This method is applicable to systems having a sufficient number of customers, adequate to produce a volume of responses suitable for analysis. The method consists of techniques offering increasing reliability with similar increase in the cost. Additionally, examples from surveys applying this method are presented. Measurements using this method and measurements using a set of internal product metrics on the same projects are compared and correlation results are discussed.

Product quality measures

According to Fenton[10], a measure is "an empirical objective assignment of a number (or symbol) to an entity to characterise a specific attribute". Therefore, surveys of customer opinion cannot be considered as measurements, since they are based on non objective assignments that vary according to user judgement. Jones[11] recognises the need to measure customers opinions and distinguish such measures (he calls them "soft data measurements") from other measurements which can be quantified with no subjectivity (he calls them "hard data measurements"). Although such hard data measurements are objective and therefore 'legal' measures, they do not actually measure product quality characteristics directly, but instead they measure internal quantities which attempt to relate to these characteristics. Unfortunately, this relation is not always successful.

McCall[12], in a classical paper, proposed a three level hierarchy model for product quality measurements. The first level consists of quality characteristics (called "factors"), the second level consists of criteria decomposing the higher level factors and the third level, the lowest level, consists of metrics being used to measure the criteria. Due to these *factors–criteria–metrics*, this model is also called FCM model. McCall proposed that low level metrics should be mapped to a set of questions that would be used in order to 'measure' each criterion. The same year, Boehm[13] also proposed a model based on a similar approach. This model was probably the basis for the international standard ISO9126[14], which was proposed many years later. The basic idea of this model is that low level metrics should be used instead of questions, in order to objectively measure attributes that are related to higher level characteristics.

The problem with all these models is their inability to combine all metrics in order to provide a global measure that will actually estimate 'software quality'. Conte[15] describes such a virtual global metric which he calls NWSC "Normalised Weighted Score". This virtual measure combines all metrics, by summing them according to weights selected by the customer. Unfortunately, such a 'super–metric' that will combine all measurements in order to provide a normalised score indicating the quality of a product, cannot exist when measuring hard data. Such a 'metric' though, can be achieved when measuring customer perceived quality using surveys. In the following section, a method that provides perceived quality measurements using such a 'metric' based on customer perceived quality surveys is presented.

The method

Surveys are a valuable tool for a quality assurance team. Modern software companies have recognised the importance of surveys and they are using both internal and external satisfaction surveys to measure from 'aspects of the employee's workplace'[16] to 'external customer satisfaction'. As argued by Kaplan[17], surveys allow focusing on just the issues of interest, since they offer complete control on the questions being asked. Furthermore, surveys are quantifiable and therefore are not only indicators in themselves, but also allow the application of more sophisticated analysis techniques appropriate to organisations with higher levels of quality maturity.

In our studies, conducted in order to measure perceived customer quality, we have used the method of mail surveys. Although surveys are valuable tools, they have to deal with four main problems:

- subjectivity of measurements,
- difficulty of statistically analysing results,
- lack of a weighing technique,
- frequency of errors

Handling the problems

Subjectivity of measurements. The simple truth is that subjectivity of measurements will remain a problem, regardless of the measurements methodology. However, the adoption and application of simple rules when planning the survey and designing the questionnaire will improve the quality of the measurements. The quality engineer who is setting up a mail survey using questionnaires must follow guidlines[18] on how to structure the questionnaire formally in order to minimise objectivity due to various interpretations of questions or choice levels. A synopsis of the guidelines we propose[19] is as follows:

- An introductory note should describe the aim of the questionnaire and the first question must be highly related to this aim. The vocabulary and phrasing must be clear and easy to understand. Explanations must be precise and brief. Furthermore, possible choices must be carefully selected and tested in order to cover all possible answers.
- The questions should be attractive to the users and the size of the questionnaire must be kept short.
- The questionnaire should be well structured and the questions must follow a logical order without references to previous questions.
- Questions with pre–defined answers should be used instead of open questions, where possible.
- Questions should be objective, to avoid leading to a specific answer (called 'halo effect') or affecting user judgement.
- Concepts such as probability which may confuse the user, should be avoided.

If these guidelines are followed during survey design, the customers will reply guided by simple rules on how to make their selection, choosing from predefined choices or even better selecting on choice bars. Therefore, problems of misunderstanding, or choosing an inappropriate answer because of subjective judgement will be minimised, resulting in measurements with increased objectivity.

Statistical analysis. As Montgomery[20] states: "Statistical methods play a vital role in quality improvement". But, the statistical method that will be used is related to the type of information contained in the measurement results. According to Yeh[21], typical measurements can be classified into one of four standard measurement scales[22]:

- Nominal scale
- Ordinal scale
- Interval scale
- Ratio scale

The problem with survey measurements is that survey data based on ordinal scale cannot be statistically analysed using formal statistical methods. This is a common problem when using questionnaires with multiple choice. No information regarding the distance between two neighbouring choices can be obtained. The solution is to use choice bars, when possible, or provide specific instructions which will explain that choices are in *interval scale*. In this case, the interviewees must fully understand that the multiple choices are in equal distance to each other.

Weighing customer opinions. In many cases (especially when measuring internal perceived quality) it is not correct to weigh all user opinions equally. Averaging survey data does not take into respect the significance of each user's opinion. Therefore, there is a need for techniques that will evaluate users' opinions according to their qualifications. The proposed techniques, as presented in the following section, take into account user qualifications and weigh user opinion based on their qualifications.

Identifying and preventing errors. Due to the nature of surveys, incorrect responses will occur. Such incorrect responses, are responses not representing user opinion and henceforth will be called 'errors'. In our surveys, we have measured a significant number of such errors caused by various factors that might seem extreme, but do occur. Such reasons are:

- The user did not answer the questionnaire himself/herself, but gave it to someone else who was inadequate to respond.
- The user answered the questionnaire very carelessly and marked randomly when he/she was confused, or just did not bother to read the instructions.
- The user started to answer with enthusiasm, but lost interest somewhere in the middle of the questionnaire and just made some random choices in order to finish it.
- The user responded with enthusiasm throughout the questionnaire but misunder-stood some questions and unintentionally provided some wrong responses.

Such errors can be prevented by following the simple rules presented previously, but cannot be eliminated. It is a challenge to design techniques that will detect such errors in order to handle answered questionnaires containing a large number of errors. Unfortunately, what Pressman[23] said about software testing, also applies here: such techniques cannot ensure the absence of errors, but they can only show that errors are present. Two of the proposed techniques presented in the following section, are used in order to detect such errors.

The techniques

The main aim of this paper is to present a rigorous approach to perceived product quality measurements that will help a quality manager to include such measurements in the company's quality assessment program. Such measurements will be carefully structured surveys to produce measurement results with a minimum degree of subjectivity, easy to analyse, respecting customer qualifications and as error–free as possible. The techniques proposed in order to measure the customers' perception of software quality are:

- QWCO
- QWCO$_S$
- QWCO$_{DS}$

These techniques are ordered with increasing reliability and increasing cost. The quality manager must select the appropriate technique according to his/her needs and apply it. **QWCO** (Qualifications Weighed Customer Opinion) is measured using the formula shown in equation (1), **QWCO$_S$** (Qualifications Weighed Customer Opinion with Safeguards) is measured using the formula shown in equation (2) and **QWCO$_{DS}$** (Qualifications Weighed Customer Opinion with Double Safeguards) is measured using the formula shown in equation (3).

$$QWCO = \frac{\sum_{i=1}^{n}\left(O_i \cdot E_i\right)}{\sum_{i=1}^{n} E_i} \tag{1}$$

$$QWCO_S = \frac{\sum_{i=1}^{n}\left(O_i \cdot E_i \cdot \frac{S_i}{S_T}\right)}{\sum_{i=1}^{n}\left(E_i \cdot \frac{S_i}{S_T}\right)} \tag{2}$$

$$QWCO_{DS} = \frac{\sum_{i=1}^{n}\left(O_i \cdot E_i \cdot \frac{S_i}{k} \cdot P_i\right)}{\sum_{i=1}^{n}\left(E_i \cdot \frac{S_i}{k} \cdot P_i\right)} \tag{3}$$

The prime aim of all these techniques is to weigh customers opinions according to their qualifications. In order to achieve this O_i, measures the normalised score of customer i opinion, E_i measures the qualifications of customer i, and n is the number of customers interviewed. Therefore, each customer contributes to the average according to his/hers qualifications.

QWCO technique, although weighs customer opinions according to their qualifications, it does not handle errors. In order to detect errors, we have proposed and used a number of safeguards embedded into the questionnaires, as shown in equation (2) representing the QWCO$_S$ technique.

Safeguard is defined as a question placed inside the questionnaire so as to measure the correctness of responses. Therefore, safeguards are not questions aiming to measure customer perceived quality, but control questions aiming to detect errors. In equation (2) S_i is the number of safeguards that the customer *i* has replied correctly to, and S_T is the total number of safeguards. Since the use of the $QWCU_S$ technique implies the use at least of one safeguard in the questionnaire, division by S_T is always valid.

Finally, $QWCO_{DS}$ technique, as shown in equation (3), uses the safeguards not only in order to detect errors when measuring customer's opinion, but also in order to detect errors when measuring customers qualifications. In equation (3), P_i value can be 0 or 1. The value of P_i is zero in case that even a single error has been detected when measuring the qualifications of customer i. P_i value is set to 1 only if the safeguards have not detected any errors while measuring the qualifications of customer i. This approach results to the rejection of a customer's responses, if errors were detected while measuring his/her qualifications. The reasoning for this approach is based on the following concept; a customer who is unreliable when answering questions regarding his/her qualifications, cannot contribute to the overall perceived software quality by having his/her opinion weighed according to such 'fake' qualifications.

Measuring customers qualifications. In order to measure customer qualifications we have presented[24] and applied a technique that allows the collection of data not only for opinion O_i of customer i for the perceived software quality, but also for the qualifications of customer *i*. Each customer is requested to fill in a set of questions requiring information for three different aspects of his/hers qualifications:

- personal background,
- syntactic knowledge,
- semantic knowledge.

Personal background is the collection of all customer qualifications which are not related to computer applications or the actual product itself. *Syntactic knowledge* is the knowledge of existing computer applications and the familiarity with the use of computers in general. According to the nature of the measured product, syntactic knowledge questions can be customised in order to inquire knowledge of specific applications related to the product. *Semantic knowledge* measures how well the customer knows the semantics of the problem automated by the product; meaning how well the customer knows the process which the software aims to facilitate.

Naturally, when measuring customer qualifications, we measure not only the knowledge, but also the years of experience of each customer. After experiments, we have assigned as default weights 0.2 for the personal background, 0.4 for the syntactic and 0.4 for the semantic knowledge. This means that personal background contributes 20% of the overall customer qualifications, syntactic knowledge contributes 40% and semantic knowledge contributes the remaining 40%. It is obvious, that the quality manager could modify these values according to the specific problem characteristics.

Measuring customers opinions. In order to measure customer opinion, we use questionnaires based on the ISO9126 international standard. Each of the six main factors of the standard (functionality, reliability, usability, efficiency, maintainability and portability) has been decomposed into a number of criteria, in a similar manner to that in McCall's model, and finally into a set of questions. The actual size of the final questionnaire, the weight of each factor and the specific criteria that will be included in order to measure each factor, vary according to the specific project requirements and depend on those quality factors on which the quality plan of the project has focused on. An example of such a question being used in order to measure the criterion "time behaviour" which will be used for the estimation of the factor "efficiency" for a database client program, is shown in figure 1.

Please rate the program's performance regarding time behaviour:

9-10 Response time equals or exceeds the requirements under all condition and resource usage.

6-8 Response time equals or exceeds the requirements in most conditions with minor limitations when resource availability is decreased. Even with these limitations the program can be used without modifications.

3-5 Response time is decreased below the acceptable limits in many situations. The system can be used but with many limitations.

0-2 Response time is decreased below the acceptable limits so often that the program cannot be used.

Figure 1 Example question from O_i measurements

In the example question illustrated in figure 1, the user is prompted to rate the program with an integer from 1 to 10. Instructions given in the description at the beginning of the questionnaire state that all responses are in interval scale and that 10 rates a perfectly satisfactory program, whereas 0 rates a completely useless program. The specific guidelines for each question are given in order to guide the user in selecting the appropriate response.

Using safeguards. For controlling the errors, in our surveys we have used three different types of safeguards embedded into the questionnaires measuring perceived product quality:

- Control Questions
- Repeated questions phrased differently
- Repeated questions offering different types of responses

Control questions are questions which can be answered only by one particular response. Any other response is an indication of error. *Repeated questions phrased differently* are questions with exactly the same meaning, but rephrased. These questions are placed into different areas within the questionnaire and have exactly the same choices as candidate answers. The selection of two different choices, no matter how distant the selected error is from the correct answer is considered an error. *Repeated questions offering different types of responses* are questions with exactly the same phrasing but with entirely different types of offered responses. In our surveys we have used such questions with multiple choice or ratio request in their first appearance and with a choice bar in their second appearance. Naturally, the second appearance has not been placed near the first. A different response to these same questions, no matter how distant the selected error is from the correct answer is considered an error. An example of a safeguard (repeated questions offering different types of responses) is shown in example questions illustrated in figures 2 and 3. These two questions were placed in two entirely different parts of the questionnaire.

Please rate the program's performance regarding ease of use:

9-10 The program can be used without any training. It attracts the user and provides a perfect working environment. On–line help is always available on any item and under any conditions.

7-8 The program can be used with minor prior training. On–line help is almost always available.

4-6 The program can be used after a training period. On–line help is generally, but not always available and in many occasions the user has to request external assistance.

1-3 The program can be used only after prior extensive training. On–line help is not provided or is totally ineffective.

0 The program is so difficult to use that in cannot be used at all.

Figure 2 First part of a safeguard

Safeguards can be used not only to preserve the integrity of answers during the survey, but also to *measure and control the effectiveness of the questionnaire structure*. We have used safeguards in the early stages of survey design, before finalising the structure of the questionnaire, in a small pilot survey with a limited number of interviewees. The purpose of this pilot survey has been to use the safeguards in order to measure the average number of errors produced when using some alternative questionnaires. These questionnaires contain the same or similar questions with alternative structures and choice types. The questionnaire which produces the minimum measured number of errors is the one selected for the final survey. Using this method we have achieved to significantly reduce the measured number of errors and therefore, to improve the overall quality of the questionnaire. Our purpose has been to be able to detect errors, but also to use the experience from this error detection phase to succeed in error prevention.

Please rate the program considering how easy it is to use. Answer by circling the response on the choice bar (select 0 if the program is so difficult to use so that it cannot be used at all, and select 10 if the program is very easy to use in a way that attracts the user and provides a perfect working environment)

|-------|-------|-------|-------|-------|-------|--------|--------|--------|--------|
0 1 2 3 4 5 6 7 8 9
10

Figure 3 Second part of the safeguard

Table 1 presents the error rates, detected by safeguards, in pilot phases from four different surveys (rows A, B, C and D). Two to four alternative questionnaire structures have been produced (columns Q_1 to Q_4) for each of the above cases, and were applied to a limited volume of interviewees. The errors measured using the safeguards on the alternative questionnaires are ordered with the worst error rate in the first column and the best in the last column. As shown in table 1, (case C) the average number of errors (detected by the safeguards) was reduced from 11.29% to 0.98%. Such a reduction, in an actual survey using 1000 interviewees, indicates that without using neither safeguards nor the pilot phase, the final survey answer sheets would have 10% higher error rate than by using the $QWCO_S$ technique. Such high error rate affects the integrity of the survey's findings and introduces a significant risk in the decision making based on the questionnaire results.

Table 1. Detected error rates in various surveys

	Q_1	Q_2	Q_3	Q_4
A	3.44%	1.37%	1.05%	–
B	6.55%	1.33%	1.08%	0.88%
C	11.29%	2.33%	0.98%	–
D	3.22%	1.20%	–	–

Another issue, using the $QWCO_S$ and the $QWCO_{DS}$ techniques, is to decide on the number of safeguards that will be used within the questionnaire. Using a great number of safeguards will cause side effects, such as increasing the size of the questionnaire and therefore causing more errors. This might result to a paradox; using safeguards to detect errors that were caused by excessive use of safeguards. Naturally, as in any real life situation, exaggerations cannot offer acceptable solutions. The manager who is responsible for the survey and the questionnaire design must decide on the number of safeguards to be used, in respect to the overall questionnaire size. In our surveys we have used a number of safeguards ranging from 5% to 10%. This number varies according to the actual questionnaire size. (In small questionnaires the percentage of safeguards used is higher than the one in large questionnaires).

Human Aspects. The use of the $QWCO_{DS}$ technique could bring up the human aspects[25] related with customer perceived software quality measurements. The use of safeguards when measuring customer qualifications could be noticed and misunderstood by the customers. This could be a major problem when such measurements are used in internal company surveys. The detection of the safeguards embedded within the questionnaire might be interpreted by the employees as an attempt to measure their qualifications which will eventually affect their career chances. This might drop the employees moral or change their attitude towards the company. Therefore, the choice of using the $QWCO_{DS}$ technique on an internal company survey is a difficult choice that the quality manager must make, taking into account all related human dimensions.

Applications of the Method

The method has been applied within the scope of our measurements program aiming to measure customer perceived quality. We have used the method in case studies[26] on a number of software projects, parallel with internal software quality measurements. For example, in the case study presented in this section, we have used an automated methodology[27] based on the 'Athena'[28] measurement environment facilitating internal software quality measurements, in order to measure internal software quality characteristics . In this case study, we have used surveys based on the $QWCO_S$ technique in order to measure the customers perception for quality.

Internal measurements were based on a triptych of commonly used internal metrics (Halstead, McCabe and Tsai[29]), which were completely automated and therefore inexpensive. The problem with internal quality measurements is that they measure internal software characteristics and not the desired external quality factors. The interpretation of the internal metrics, in order to estimate these factors, is difficult and not always successful. External measurements (customer perceived quality measurements) for the purpose of this case study, were based on surveys. These kind of measurements have a higher cost level than internal measurements, but offer results on customers perception for the desirable external quality characteristics.

Table 2 shows the normalised measurement results of the 46 software products measured using the $\mathrm{QWCO_S}$ technique, in assenting order (worst measurements first). These projects were the measured samples for the presented case study. The measurements of customer perception of quality were based on a total of 1551 responded questionnaires with proper product evaluation from various users. Table 3 shows the internal measurement results for the same 46 projects in the same order as in table 2. The internal measurement results are normalised and derived using a combination formula for the metrics implemented into the internal measurements program. This combination metrics formula (CMF) does not measure a physical quantity of the product, but combines all metric results. Its solid purpose is to provide a collective mechanism for comparison as shown in equation (4).

Table 2. $\mathbf{QWCO_S}$ measurements

0.06	0.12	0.12	0.14	0.16	0.19	0.19	0.21
0.23	0.26	0.28	0.30	0.30	0.30	0.31	0.33
0.33	0.34	0.34	0.39	0.40	0.41	0.42	0.43
0.43	0.44	0.45	0.46	0.47	0.48	0.48	0.48
0.51	0.54	0.56	0.60	0.60	0.60	0.67	0.75
0.76	0.78	0.80	0.86	0.88	0.94		

Table 3. Normalised CMF measurements

0.29	0.48	0.77	0.37	0.22	0.36	0.61	0.35
0.39	0.33	0.52	0.35	0.75	0.40	0.51	0.58
0.49	0.46	0.50	0.80	0.39	0.70	0.66	0.47
0.64	0.75	0.48	0.47	0.49	0.53	0.55	0.47
0.57	0.60	0.61	0.52	0.72	0.62	0.98	0.72
0.68	0.91	0.75	0.88	0.87	0.89		

The metrics that were chosen to participate in the CMF are the weighed average language level (λ_{wa}), the essential size ratio (R), the weighed average cyclomatic complexity (V_{wa}) and the data structure complexity metric (T). As an indication of the language level for the entire project, the weighed average language level, which is shown in equation (5), was used in order to measure the contribution of the language level of every routine *i* into the overall project language level. A project is a collection of routines created by various programmers. Each one of these routines has a different language level and contributes to the project language level according to the routine's size.

$$CMF = 0.2 \cdot \lambda_{wa} + 0.2 \cdot R + 0.4 \cdot V_{wa} + 0.2 \cdot T \quad (4)$$

$$\lambda_{wa} = \frac{\sum_i \left(N_i \cdot \lambda_i \right)}{\sum_i N_i} \quad (5)$$

The use of the essential size ratio R, which is measured as shown in equation (6), is justified by the analyses[30,31] indicating that $N^{\wedge}$ measures the optimal module size without any code impurities. Therefore, R provides an indication of the proper, or not, use of the programming language.

$$R = \frac{N^{\wedge}}{N} \quad (6)$$

In a similar manner to λ_{we}, the cyclomatic complexity weighed average (V_{wa}) is the ratio of 10 by the weighed average of McCabe's metric for each routine *i*. The number 10 is the proposed highest acceptable complexity by McCabe. The formula to measure V_{wa} is shown in equation (7). Finally, in order to measure the data structures complexity T, the higher polynomial exponent (T_{ex}) from the derived data structure polynomials was used as shown in equation (8). We must emphasise again that proper analysis is based on the individual results of all metrics, but since the complete set of results for all the metrics used is not easy to present in a paper, we use CMF which provides a way to combine all metrics in a easy to present manner.

$$V_{wa} = 10 / \left(\frac{\sum_i \left(N_i \cdot Vg_i \right)}{\sum_i N_i} \right) \quad (7)$$

$$T = \frac{1}{T_{ex} + 1} \quad (8)$$

The correlation between the measurement results of tables 2 and 3 was measured to be 70.44%. Such correlation shows that the internal measures used, do not completely conform to the customer perceived quality measurements. The scatter plot of figure 1 illustrates this correlation by representing the $\mathrm{QWCO_S}$ measurements in the horizontal bar and the normalised CMF results in the vertical bar. The diagonal line is the correlation line, representing the line where all point should be if the two measurement methods were 100% correlated. The points which are marked below this correlation line represent projects that, although they have not satisfied internal measurement standards, they have achieved higher than expected scores in customer perceived quality measurements. The points marked above the correlation line represent projects that, have satisfied internal measurement standards, but have not achieved equally high scores in customer perceived quality measurements.

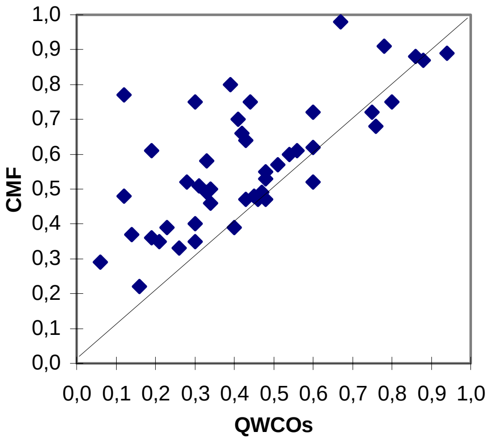

Figure 4 Scatter Plot of 46 projects

As one can see in the scatter plot, there are almost no projects which fail the internal quality measurements and achieve high scores in customer perceived measurements. Very few points are below the correlation line and no points are in great distance below this line. On the contrary, many points are not only just over the correlation line, but further above. We must emphasise the fact that all the individual measurements for each metric used, produced similar scatter plots. The CMF is used only to serve as a collective formula for presentation purposes. This can lead us to the conclusion that, although internal metrics can detect programs that might get low scores in terms of customers perception for their quality, satisfaction of internal measurement does not guarantee achievement of high scores in customer perceived quality measurements. Keeping in mind that customer perception of quality is a success measure for software companies, we can conclude that internal metrics offer a great means for detecting programs that might cause low customer perceived quality measurements. Naturally, this is not the only purpose of internal measurements. However, since internal measurements cannot fully detect programs that will have low customer perceived quality measurements, the use of surveys is required in order to measure the actual customer perceived quality and in order to test and calibrate the internal measurement procedures.

Conclusion

This paper presents a method which focuses on the definition of software quality as ‘satisfaction of customer requirements’. This method fits into any quality assurance framework and especially to those based on ISO9000, IEEE, or Baldrige. As any method, it has advantages and disadvantages. The disadvantages are cost in deploying the techniques, error rates, subjectivity of the answers and human factors involved with surveys and qualification measurements. This paper offers solutions in the form of techniques and guidelines in order to overcome these disadvantages. The quality manager must weigh the priorities for each specific case and decide which one of the proposed techniques will be used.

The main advantages of the method presented in this paper are: a) it conforms with the definition of quality, b) it fits in almost every quality assurance framework, c) it is quantifiable; it measures directly external quality factors and can be subject to more sophisticated analysis techniques, appropriate to organisations with higher levels of quality maturity, d) it is always applicable and does not depend on programming languages or tools and e) it offers interaction with customers thus providing confidence that the company respects their opinion.

The application of this method in parallel with software quality measurements based on internal metrics, proves that the method can actually be used in parallel with such measurements. The aim of this method is not to substitute internal metrics, but to offer an alternative solution. This solution emphasising that quality focuses on customer requirements, should be used in parallel with current practices in order to aid in calibrating metrics, in controlling measurement results, and in providing confidence to both the company and the users.

Acknowledgements

The authors would like to thank the anonymous referees for their careful review and suggestions and Ms. M. Stamison–Atmatzidi, EFL Instructor at the Dep. of Computer Engineering and Informatics for the proof-reading and corrections of the English.

References

1 ISO, ‘Quality Management and Quality Assurance Standards’, International Standard, ISO/IEC 9001: 1991
2 Ince, D, ‘ISO 9001 and Software Quality Assurance’, Quality Forum, McGraw Hill, isbn: 0-07-707885-3, 1994
3 IEEE, ‘Standard for a Software Quality Metrics Methodology’, P-1061/D20, IEEE Press, New York, 1989
4 Brown, M G, ‘Baldrige Award Winning Quality: How to Interpret the Malcom Baldrige Award Criteria’, Milwaukee, WI: ASQC Quality Press, 1991
5 Steeples, M M, ‘The Corporate Guide to the Malcom Baldrige National Quality Award’, WI: ASQC Quality Press, 1993
6 Albrecht, A J, ‘Measuring application development productivity’, Proc. of IBM Apllic. Dev. Joint SHARE/GUIDE Symposium, Monterey, CA, pp. 83-92, 1979
7 McCabe, T J, ‘A complexity measure’, IEEE Trans. Soft. Eng. SE-2(4), pp. 208-320, 1976
8 Halstead, M H, ‘Elements of Software Science’, Elsevier North Holland, 1977
9 Kent, R, ‘Marketing Research in Action’, Routledge, London, isbn: 0-415-06759-6, 1993
10 Fenton, N E, ‘Software Metrics A Rigorous Approach’, Chapman & Hall, isbn: 0-442-31355-1, 1992
11 Jones, C, ‘Applied Software Measurement: Assuring Productivity and Quality’, McGraw Hill, isbn: 0-07-032813-7, 1991

12 McCall, J A, Richards, P K, and Walters, G F, 'Factors in Software Quality, Vols I, II, III', US Rome Air Development Center Reports NTIS AD/A-049 014, 015, 055, 1977

13 Boehm, B W, Brown, J R, Kaspar, J R, Lipow, M, McCleod, G J, and Merrit, M J, 'Characteristics of Software Quality', North Holland, 1978

14 ISO, 'Information technology - Evaluation of software - Quality characteristics and guides for their use', International Standard, ISO/IEC 9126: 1991

15 Conte, S D, Dunsmore, H E, and Shen, V Y, 'Software Engineering Metrics and Models', Benjamin Cummings, isbn: 0-8053-2162-4, 1986

16 Evvardsson, B, Thomasson, B, and Ovretveit, J, 'Quality of Service. Making it Really Work', McGraw Hill, isbn: 0-07-707949-3, 1994

17 Kaplan, C, Clark, R, and Tang, V, 'Secrets of Software Quality', McGraw Hill, isbn: 0-07-911795-3, 1995

18 Lahlou, S, Van der maijden, R, Messu, M, Poquet, G, and Prakke, F, 'A Guideline for Survey – Techniques in Evaluation of Research', Blussels, ESSC–EEC-EAEC, 1992

19 Xenos, M, and Christodoulakis, D, 'Evaluating Software Quality by the Use of User Satisfaction Measurements', 4th Software Quality Conference, SET, University of Abertay, Dundee, pp. 181-188, 1995

20 Montgomery, D C, 'Introduction to Statistical Quality Control', second edition, John Wiley & Sons, isbn: 0-471-51988-X, 1991

21 Yeh, H T, 'Software Process Quality', McGraw Hill, isbn: 0-07-072272-2, 1993

22 Stevens, S S, 'On the Theory of Scales of Measurement', Science, 103: 677, 1946

23 Pressman, R S, 'Software Engineering. A Practitioner's Approach', 3rd edition, McGraw Hill, isbn: 0-07-050814-3, 1992

24 Xenos, M, and Christodoulakis, D, 'Software Quality: The User's Point of View', pp. 266-272 of Software Quality and Productivity, Chapman & Hall, isbn: 0-412-62960-7, 1995

25 Thomas, B, 'The Human Dimension of Quality', McGraw Hill, isbn: 0-07-709051-9, 1994

26 Xenos, M, Stavrinoudis, D, and Christodoulakis, D, 'The Correlation Between Developer-oriented and User-oriented Software Quality Measurements (A Case Study)', 5th European Conference on Software Quality, EOQ-SC, Dublin, pp. 267-275, 1996

27 Xenos, M, and Christodoulakis, D, 'An Applicable Methodology to Automate Software Quality Measurements', IEEE Software Testing and Quality Assurance International Conference, New Delhi, pp. 121- 125, IEEE ID: 0-7803-2608-3, 1994

28 Tsalidis, C., Christodoulakis, D., and Maritsas, D., 'Athena: A Software Measurement and Metrics Environment', Software Maintenance Research and Practice, 1991

29 Tsai, W. T., Lopez, M. A., Rodriguez, V., and Volovik, D., 'An Approach to Measuring Data Structure Complexity', Compsac86, pp. 240–246, 1986

30 Fitzsimmons, A., and Love, T., 'A Review and Evaluation of Software Science', Computing Surveys, Vol. 10, No 1, pp. 45-60, 1978

31 Christensen, K., Fitsos, G. P., and Smith, C. P., 'A Perspective on Software Science', IBM Syst. Journal, Vol. 20, No 4, pp. 372- 387, 1986