Quantification of Software Requirements Specification and Software Quality: Quantitative Approach

Wesen Tegegne
Commercial Bank of Ethiopia,
Addis Ababa, Ethiopia
wesenteg@gmail.com

Mesfin Belachew
Ministry of Communication and IT, Addis Ababa, Ethiopia
mesfinbelachew@gmail.com

Mesfin Kifle
Department of Computer Science,
Addis Ababa University, Ethiopia
kiflemestir95@gmail.com

Abstract

One of the major problems of local software development firms and organizations is absence and unavailability of developed quantitative quality factors measurement technique that could assist to rate Software Requirements Specification (SRS) and software quality. In this paper a quantitative quality measurement, Quality Quantification Technique (QQT), is proposed by adapting and adopting from available measurements to quantify quality of SRS at requirement stage of System Development Life Cycle (SDLC) and software product. It provides 14 steps.

QQT uses propositional parameter for measuring quality attributes using linguistic variables. Due to human decision-making where the decisional criteria are not clear-cut, but blurred, it attempts to quantify the quality using fuzzy logic that can be defuzzified to get the final quality in crisp value. In QQT, application of fuzzy logic technique in measuring quality helps to resolve vagueness that could exist to some extent. The technique used in quantification of software quality is adopted and adapted to establish the base for SRS quality measurement. In addition to adopting and adapting the available software metrics fuzzy value, we tried to incorporate self-understanding and experience of inferring quality attributes. In order to evaluate the proposed QQT, a case study is carried out to measure SRS and software quality.

Keywords: SRS Quality, Software Quality, Fuzzy Logic, Defuzzified, QQT

1. Introduction

Looking at software engineering from a historical perspective, the 1990s and beyond could be viewed as the quality and efficiency era. In this era, demand for quality is further intensified by the ever increasing dependence of society on software. Quality has been brought to the center of the software development process. Software development has to be more efficient and the quality level of the delivered products has to be high to meet requirements and to be successful. These factors will continue to affect software engineering for many years to come [1].

The need for quality measurement is increased by the rising reliance of society on software. This is where the demand for measuring quality becomes apparent, be it qualitative or quantitative. In qualitative measurement, quality is measured in a more subjective manner whereas in quantitative measurement, the focus is in establishing metrics that could appropriately measure the degree of quality. In this paper, a quantitative technique that uses fuzzy logic to measure quality of SRS and software is applied.

2. Background

SRS Quality: Writing a good requirements document is difficult in the eyes of most developers. It delays getting on with the coding and it is thus considered a waste of time [2]. If we can better understand how to recognize and measure quality in SRS, we will be better equipped to detect errors in the SRS. SRS errors need to be detected during the requirements phase, or the cost to repair them will grow significantly [3]. As Davis *et al.* [3] quoted from Daly, Bohem *et al.* and Fagan they provide conclusive evidence that the later in the lifecycle an error is detected and repaired, the more it will cost. The ratio between detecting and repairing an error during requirements versus maintenance phases is 200:1

Software Quality: According to Herb [4], even though many successful software products and systems exist in the world today, an overall lack of attention to quality has also led to many problematic systems that do not work right as well as to many software projects that are late, over budget, or canceled "… so many systems are delivered that are not used, or not liked … we all believe we have done a good job and are mortified if our work is not well received [5]". Identifying quality requirements that can be elicited, formalized, and further evaluated in each phase of full software product lifecycle becomes a crucial task in the process of building a high quality software product [6].

According to Challa *et al.* [7], fuzzy logic provides an easier way to infer definite conclusions from highly imprecise, vague, and ambiguous information when compared with classical logic and relies on the experience of the operator rather than the technical understanding of the subject.

Applying fuzzy logic for the quantification of SRS and software quality is the basic concept behind fuzzy logic. According to Changli [8], fuzzy logic is inspired by the human processes for quality measurement where the decisional criteria are not clear-cut, but blurred, and it is difficult to find objective measurement to make the decisions more precise and clear. In this quantitative quality measurement, fuzzy logic is used. According Zadeh [9], fuzzy logic is a precise logic of impression and approximate reasoning.

3. Related Work

For some quality factors of SRS, Davis *et al.* [3] tried to give objective measurement, recommended weights of importance and reference values for attributes. They suggested that some of SRS qualities appear to be essential for all applications and gave weights of 1. Others seem less important in general and gave lower weights. The actual weights for all the attributes must be assigned by each project to be meaningful. They proposed SRS quality equation, $Q = \Sigma W_i Q_i / \Sigma W$ where W is weight of quality factors and i is set of SRS quality attributes such as {complete, correct, etc.}. Some of the limitations are most quality attributes have no objective measurement and weight of importance and reference values for attributes are not suggested.

Bawane *et al.* [10] proposed a quantitative model to establish the quality requirements expected by various stakeholders and to incorporate these requirements in the product under development. The quantitative quality model takes a set of quality requirements as input for the development of a software application. Some of the limitations are that it is model specific and does not apply methods to solve vagueness and imprecision which exist during subjective judgments of relative priorities of quality characteristics by stakeholders.

The authors in [11] identified software quality factors for the telecommunication industry in Malaysia. The quality characteristics and sub characteristics of a product were isolated from literature review to reflect what is currently thought to be important. Then they are converted into meaningful questions in a questionnaire and sent to stakeholders (users, developers, and managers) who expressed their quality opinions. Then, they gathered the questionnaires and consolidated the opinions from the correspondents. Some of the limitations are that it is model specific and there exist subjective judgments in determining relative priorities of quality characteristics by stakeholders.

Challa *et al.* [7] attempted to quantify software quality factors with the help of quality factors stated in ISO 9126 model. The software quality model is sub divided into perspectives, characteristics, sub characteristics, and metrics. Three perspectives (users, developers, and managers) are used to express their quality opinions. The project manager's perspective has been separately added to the model. Each perspective is sub divided into various characteristics. The sub characteristics included in the project manager's perspective are: cycle time, cost, and schedule pressure.

Every characteristic is further sub divided into sub characteristics and every sub characteristic is further sub divided into metrics. At these levels, parameter is associated with a corresponding rating and weight. Fuzzy weighted average of the metrics is taken to evaluate the rating of the sub characteristic. Then fuzzy weighted average of the characteristics is taken to get the rating of the perspective. Fuzzy weighted average of different perspectives is taken to get the final software quality in terms of a fuzzy set. Centroid formula is then employed on this triangular fuzzy set to calculate the final software quality [10]. Some of the limitations are that it does not provide a technique to include other stakeholders for quality measurement and it is not generic (based on ISO 9126 model).

Our proposed solution to quantify quality is a remedy to the limitations in [3, 7, 10]. In our proposed solution, different stakeholders' views are included to express their quality opinions. Quantification techniques include SRS quality (not only for software quality), it is not to be quality model specific (the model is a general one). In SRS quality measurements, fuzzy logic is applied to solve vagueness and imprecision that could exist during subjective judgments. A general formula is given for quantification of net quality and the quantification process is general (not specific to software quality only). Project managers' characters (cost, time, and schedule pressure) is added separately to measure quality of SRS. We also suggest measurement techniques for attributes that have no objective measurement.

4. The Proposed Solution

All the steps required to follow in quality quantification process are listed in Table 1 and can be used as a checklist. QQT has three main stages: quality analysis, quality design, and quality measurement. In turn, each of these stages has further sub steps. The main objective of quality analysis is to identify and include the view points of different stakeholders and their quality requirements based on their main interest or the definition of quality they favor. In managing a software project, stakeholders could have different views or expectation in measuring quality attributes.

Quality design is the stage that is going to prepare metric questions for stakeholder quality characteristics and preparation of format to be filled out by stakeholder information. QQT will use fuzzy logic. The weighted average technique of triangular fuzzy sets is used. Once all the stakeholders' quality requirements for SRS or software are identified and gathered for each group, each attribute will be quantified by individual metric. Linguistic variables for the ratings and weights of the fuzzy sets for the quality metrics defined as VL (Very Low), L (Low), M (Medium), H (High), and VH (Very High).

Table 1: The Fourteen QQT steps

1	Identify stakeholders	Quality Analysis
2	Classify stakeholders	
3	Identify stakeholders' quality requirement	
4	Specify metrics for each quality characteristics	Quality Design
5	Define fuzzy value for quality attribute	
6	Determine ratings of metrics for quality attribute	
7	Determine weight of metrics for quality attribute	
8	Define the weights of the fuzzy sets for quality metrics	Quality Measurement
9	Define the ratings of the fuzzy sets for quality attribute	
10	Compute average fuzzy ratings of quality metrics	
11	Compute weighted average of quality	
12	Compute average weight of quality	
13	Compute net quality	
14	Defuzzification	

In order to measure the quality factors of SRS and software, triangular fuzzy set is used for simplicity. For every metric, there will be a corresponding rating and weight. The rating and the weight of any metric is fuzzified into triangular fuzzy sets. The required inputs of the ratings of the metrics and weights at different levels will be obtained from stakeholders via questionnaire or in an interactive discussion with stakeholders. The weighted average technique is used to calculate the final net quality. The pictorial representation of weighted average of quality process in shown in Figure 1. In the process of calculation, levels could be omitted if a quality factor does not have sub characteristics. Moreover, the levels can be extended if the classification hierarchy is increased.

Let classification be the name given to the levels in quantification process. These are quality (level 0), perspectives (level 1), character (begins at level 2), sub characteristic (ends at level N-1), and metrics (level N) classification names. The fuzzy rating of the Level (N-M)+1 classification is obtained by a weighted average of Level N- M classification affecting it. It can be written as a formula.

Rating of Level N-M classification = $r_1 \times w_1 + r_2 \times w_2 + \ldots r_n \times w_n = \sum r_i \times w_i$

where r is fuzzy average rate of Level (N-M)+1 classification, w is fuzzy average weight of the Level (N-M)+1 classification, and i belongs to the set of Level (N-M) +1 classification affecting Level (N-M) classification, M = 1, …, N.

The process of converting a real time problem into fuzzy set is called fuzzification. The last step in QQT is defuzzification which is the process of converting the fuzzy sets obtained during fuzzification process into crisp or real time data. The fuzzy ratings of the quality with respect to different perspectives and the net quality are defuzzified. The centroid method is adopted to defuzzify the triangular fuzzy sets [7].

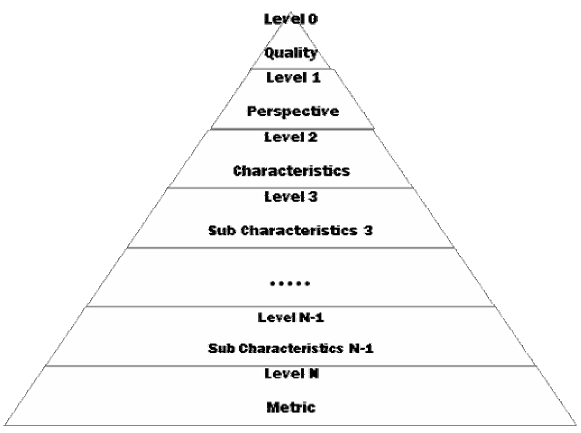

Figure 1: Weighted average calculation process

Centroid method $Z^* = \dfrac{\int \mu(z) z dz}{\int \mu(z) dz}$

Here z* is the defuzzified crisp value, z is the value on the X axis, and μ(z) is the membership function.

The inference table for SRS and software quality is shown in Table 2 which is adopted from [7]. The quantified quality level lies in the range of 0 to 1. Calculated quality value that is closer to zero indicates the poorest quality level and a value closer to one shows excellent quality.

Average quality level might be acceptable for non-critical systems if it could not compromise the major functionalities of the software. However, it is highly recommend having at least good quality level for non-critical systems. If software quality is below average (poor or very poor), it is unacceptable.

Table 2: Inference on Quality

Overall Quality Calculated	*Inference on Quality*
More than 0.65	Very Good
Between 0.5 and 0.65	Good
Between 0.35 and 0.5	Average
Between 0.25 and 0.35	Poor
Less than 0.25	Very Poor

Software quality always depends on the context in which software operates. Good quality level might be dangerous depending on the environment in which the software operates. A software application that is critical to the operations of the organization will require very good quality level. A higher level of quality even closer to one might be important.

SRS is the success of all software projects and it is the basis for all subsequent stages of SDLC. Very good quality level for SRS could lead to excellent software. If the quality level of SRS is close to one, then the cost in detecting and repairing an error in the later life cycle of software development will decrease significantly. Hence, SRS quality is acceptable if its quality level is at least very good and a level of quality is preferable if it is closer to one.

5. Case Study

Company A has prepared SRS and this is to evaluate SRS quality on the basis of the selected quality attribute from Davis *et al*. [3], Wiegers [6], and IEEE [12]. Besides, project manager's perspective quality factors (cycle time, cost, and schedule pressure) are added to the measurement. Fourteen QQT steps are used as check list for measurement. Stakeholders are classified based on the definition of the quality they favor and their main interest. They are builders, measurers, and managers. It is agreed and negotiated with compromise that weight of importance to builders perspectives are very high, managers perspectives are high, and measurers perspectives are average in the SRS quality.

Table 3: Fuzzy triangular number for the weights

Importance of Criteria	*Fuzzy Weights*
Very Low	(0.0, 0.0, 0.25)
Low	(0.0, 0.25, 0.5)
Medium	(0.25, 0.5, 0.75)
High	(0.50, 0.75, 1.0)
Very High	(0.75, 1.0, 1.0)

Table 4: Fuzzy triangular number for the ratings

Ratings of Criteria	*Fuzzy Ratings*
Very Low	(0.0, 0.1, 0.3)
Low	(0.1, 0.3, 0.5)
Medium	(0.3, 0.5, 0.7)
High	(0.5, 0.7, 0.9)
Very High	(0.7, 0.9, 1.0)

The characteristics are identified from literature review and discussion among stakeholders to reflect what is currently thought important to SRS. Stakeholders filled out different questionnaire. Three builders, two measurers, and two managers filled out questionnaire of input for SRS measurement. The fuzzy value of weights and ratings are shown in Tables 3 and 4, respectively.

Now the fuzzy rating of the overall SRS quality (net quality) is calculated using

r Net Quality = r Builder's×w Builder's+ r Measure's× w Measure's+ r Project Manager's× w Project Manager's. r Net Quality = (.33,.79,1)

Centroid method $Z^* = \dfrac{\int \mu(z) z dz}{\int \mu(z) dz}$

So the crisp value of the SRS net quality is calculated as 0.70.

Table 5: Fuzzy ratings for net SRS quality and quality with respect to different perspective

Net SRS Quality	*Quality*	*Net Weighting*	*Net Rating*	*Quality Attribute*	*Average Fuzzy Weight*	*Average Fuzzy Rating*
(0.33,0.79,1)	Builder	(0.75, 1.0, 1.0)	(.44, .79, 1)	Achievable	(0.75,1,1)	(0.63, 0.83,0.97)
				Annotated	(0.33,0.58,0.83)	(0.7,0.9,1)
				Complete	(0.25,0.5,0.75)	(0.7,0.9,1)
				Consistent	(0.75,1,1)	(0.7,0.9,1)
				Correct	(0.67,0.92,1)	(0.7,0.9,1)
				Modifiable	(0.33,0.58,0.83)	(0.23,0.43,0.63)
				Necessary	(0.42,0.67,0.92)	(0.5,0.7,0.9)
				Precise	(0.58,0.83,0.92)	(0.3,0.5,0.7)
				Reusable	(0.17,0.33,0.58)	(0.43,0.63,0.8)
				Traceable	(0.75,1,1)	(0.3,0.5,0.7)
				Unambiguous	(0.5,0.75,0.83)	(0.1,0.3,0.5)
				Understandable	(0.67,0.92,1)	(0.5,0.7,0.9)
	Measure	(0.25, 0.5, 0.75)	(.44, .79, 1)	Achievable	(0.63,0.88,1)	(0.5,0.7,0.85)
				Annotated	(0.13,0.38,0.38)	(1,0.3,0.25)

Net SRS Quality	*Quality*	*Net Weighting*	*Net Rating*	*Quality Attribute*	*Average Fuzzy Weight*	*Average Fuzzy Rating*
				Complete	(0.25,0.5,0.75)	(0.7,0.9,1)
				Consistent	(0.25,0.5,0.75)	(0.5,0.7,0.85)
				Correct	(0.63,0.88,1)	(0.6,0.8,0.95)
				Modifiable	(0,0.25,0.25)	(0.65,0.4,0.35)
				Necessary	(0.38,0.63,0.88)	(0.4,0.6,0.8)
				Precise	(0.5,0.75,0.88)	(0.4,0.6,0.8)
				Reusable	(0,0.13,0.38)	(0.65,0.4,0.6)
				Traceable	(0.25,0.5,0.75)	(0.6,0.8,0.95)
				Unambiguous	(0.38,0.63,0.75)	(0.7,0.9,1)
				Understandable	(0.63,0.88,1)	(0.7,0.9,1)
	Manager	(0.50, 0.75, 1.0)	(.35, .68, 1)	Cycle Time	(0.38,0.63,0.88)	(0.5,0.7,0.9)
				Cost	(0.13,0.38,0.25)	(0.3,0.5,0.7)
				Schedule Pressure	(0.5,0.75,1)	(0.7,0.9,1)

Based on inference quality Table 5, and its SRS quality interpretation, since the net SRS calculated value is 0.70, we can say that the SRS has very good quality and it is acceptable. Due to space limitation, this paper does not elaborate each of the 14 steps of QQT and the case study carried out to measure software quality. For detailed information, please refer to [13].

Table 6: Crisp value of perspectives' SRS quality

Name of Perspective	*Perspective quality*	*Crisp value*
Builders	(.44, .79, 1)	0.74
Measurers	(.44, .79, 1)	0.74
Managers	(.35, .68, 1)	0.67
*Total Quality (*net quality)	(.33, .79, 1)	0.70

6. Conclusion

In this paper, a quantitative SRS and Software quality measurement technique, QQT, is proposed. Software development firms and organizations or stakeholders can utilize QQT to measure quality status of the SRS at requirement stage of SDLC and the end result of software quality quantitatively. QQT is also useful for developers, users, and project managers to measure quality from each perspective and net quality. The proposed technique is easy and understandable. It uses propositional parameter for measuring quality attributes using the linguistic variables very low, low, medium, high, and very high.

These linguistic variables are fuzzy because the boundary of variables is not clear-cut. It is difficult to find objective to make the decisions more precise and clear. QQT uses fuzzy logic due to human decision making where the decisional criteria are not clear-cut, but blurred. Fuzzy logic uses to analyze approximate data to precise solutions. In QQT, the quality parameters are quantified in terms of fuzzy set and finally converted to crisp or numeric value.

The fuzzy value or criteria to metrics are not without dispute. The approach uses experience and opinion of a person involved in the process of measurement. Different persons can set fuzzy values for fuzzy metrics differently. The other limitation is also that quality classification among perspectives by different persons may not be the same.

References

[1] Stephen Kan H., "Metrics and Models in Software Quality Engineering", Addison Wesley, 2nd edition, 2002.

[2] Tom Glib and Lindesy Brodie, "What's Fundamentally Wrong? Improving Our Approach Towards Capturing Value in Requirements Specification", http://www.

requirementsnetwork.com/system/files/Whats Fundamentally Wrong.pdf, last accessed on August 06, 2011.

[3] Alan Davis, Scott Overmyer, Kathleen Jordan, Joseph Caruso, Fatma Dandashi, Anhtum Dinh, Ledeboer Kincaid, Gary, Glen, Patricia Reynolds, Pradip Sitaram, Anh Ta, and Mary Theofano, “Identifying and Measuring Quality in a Software Requirements Specification”, First International Software Metrics Symposium, Proceedings, 1993.

[4] Krasner, Herb. “Using the Cost of Quality Approach for Software”, Crosstalk Journal of Defense Software Engineering, 1998.

[5] Isabel Evans, “Achieving Software Quality through Teamwork”, Norwood, 2004.

[6] Wiegers, Karl E. “Software Requirements”, Microsoft Press, 2003.

[7] Jagat Sesh Challa, Arindam Paul, Yogesh Dada, Venkatesh Nerella, Praveen Ranjan Srivastava, and Ajit Pratap Singh, “Integrated Software Quality Evaluation: A Fuzzy Multi-Criteria Approach”, Journal of Information Processing Systems, Vol. 7, No. 3, 2011.

[8] Sun Changli, “Software Documents Quality Measurement - A Fuzzy Approach”, Information Theory and Information Security (ICITIS), IEEE International Conference, 2010.

[9] Loti A. Zadeh, “Is there a need for Fuzzy Logic?”, Fuzzy Information Processing Society, NAFIPS.

[10] Bawane, Neelam and Srikrishna C. V, “A Novel Method for Quantitative Assessment of Software Quality”, International Journal of Computer Science and Security, Volume 3: Issue 6, 2009.

[11] Nor Fazlina, Iryani Abdul Hamid, and Mohamad Khatim Hasan, “Identifying Software Quality Factors for Telecommunication Industry in Malaysia”, International Conference on Electrical Engineering and Informatics, 17-19 July 2011.

[12] IEEE Standard Board, IEEE Recommended Practice for Software Requirements Specifications, IEEE, 1998.

[13] Wesen Tegegne, “Software Requirements Specification and Software Quality Measurement: Quantitative Approach”, Unpublished Master’s Thesis, HiLCoE School of Computer Science and Technology, 2012.