2010 Fourth Asia International Conference on Mathematical/Analytical Modelling and Computer Simulation

A Quality-Based Requirement Prioritization Framework Using Binary Inputs

Carlos E. Otero, Erica Dell, Abrar Qureshi
Department of Mathematics and Computer Science
University of Virginia - College at Wise
Wise, VA USA
cotero@virginia.edu

Luis D. Otero
Department of Engineering Systems
Florida Institute of Technology
Melbourne, FL USA
lotero@fit.edu

***Abstract*— Despite the clear need to prioritize requirements in software projects, finding a practical method for requirements prioritization has proven difficult. Existing requirements prioritization methods that provide the most consistent results are also the most complex, and therefore the most difficult to implement. More informal methods save time and are easier to apply, but may not be suitable for practical scenarios because they lack the structure and consistency required to properly analyze requirements. This paper proposes a novel and practical approach for prioritizing requirements in software projects. The proposed approach attempts to quantify the quality of requirements to provide a measurement that is representative of all quality criteria identified for a specific software project. The derived quality measurement can be easily computed to serve as the main metric for requirements prioritization.**

Keywords: Requirements Engineering, Requirements Prioritization, Desirability Functions, Software Engineering, Software Quality,Software Process

I. INTRODUCTION

Software is continuing to become an increasingly integral part of day-to-day life. Its presence is ubiquitous; people rely on software for a myriad of purposes, from controlling safety systems in automobiles to recreation. As the prevalence of software increases, so does the complexity, as well as the number of requirements that are derived for modern software projects [1]. This presents a dilemma for program managers and software engineers, who must complete projects given a finite amount of resources and time. Inevitably, some requirements cannot be fulfilled if a project is to be completed on schedule. In order to ensure that the most important requirements are implemented, it is essential that requirements be prioritized appropriately. Project managers, customers, and other stakeholders must determine the benefit and costs associated with each requirement, and establish their relative importance.

Despite the clear need to prioritize requirements, finding a simple and effective method for requirements prioritization has proven difficult. Two important factors associated with requirements prioritization are the benefit and cost of each requirement [2]. However, according to Herrmann et al., the techniques studied in their systematic review (SR) use a process designed to estimate the benefit of entire systems as opposed to the benefits of individual requirements. In fact, they reported that, "We found no methods which estimate benefit for individual requirements." [2].

Another challenge to requirements prioritization is that many existing methods are too complex to be implemented by software organizations. According to [3], project managers still lack access to requirements prioritization techniques that are sufficiently effective, as well as practical. There appears to be, in existing requirements prioritization methods, a tradeoff between consistency and ease of use. The methods that provide the most consistent results are also the most complex, and therefore the most difficult to implement. More informal methods save time and are easier to apply, but are not suitable for complex projects because they lack the structure and consistency required to properly analyze a complex set of requirements [4]. In order to surmount these challenges and facilitate widespread adoption of requirements prioritization techniques, more practical methods must be devised.

This paper proposes a novel approach for prioritizing requirements in software projects. The proposed approach attempts to quantify the quality of requirements to provide a measurement that is representative of all quality criteria identified for a specific software project. The derived quality measurement can be used as the main metric for requirements prioritization. The remainder of the paper is organized as follows. Section II provides a brief summary of previous work on requirements prioritization. Section III provides a brief summary of the solution approach. Section IV provides detailed explanations of the desirability functions technique. Section V presents the results of a case study. Finally, Section VI provides summarized conclusions and highlights of the proposed approach.

II. BACKGROUND WORK

As software has become more complex, and project managers are forced to make concessions and trade-offs to complete projects on schedule, requirements prioritization has become an increasingly important part of ensuring the success of a project. There are many compelling arguments as to why requirements prioritization is necessary. One of the most compelling is made by Karl Wiegers. He argues that limited resources inevitably mean that some

978-0-7695-4062-7/10 $26.00 © 2010 IEEE
DOI 10.1109/AMS.2010.48

requirements cannot be implemented, and that the decisions about which requirements are the most important are better made in early development stages rather than in "emergency mode" towards the end of a project [4].

Most requirements prioritization methods (RPM) involve examining requirements through the framework of benefit and cost [2]. In other words, requirements are analyzed on the basis of how much benefit that fulfilling the requirement will provide to the customer, as well as any costs associated with its implementation. This information is then used in some manner to rank the requirements in terms of their importance.

There are a number of methods that currently exist for approaching requirements prioritization. Many of these methods are quantitative, and employ a very systematic approach to gathering data and assigning values to various factors associated with requirements in order to compute a priority [2]. Other methods rely on making somewhat informal generalizations and groupings before trying to assign priorities. This is typically done to reduce the amount of time necessary to compute priorities, but may sacrifice some consistency [5].

One of the most consistent methods that have been developed is the Analytic Hierarchy Process (AHP)[3]. All possible pairs of requirements are enumerated, and then the perceived importance of each requirement is ranked in relationship to its pair. The most important requirement from each pair is assigned a value, while the requirement of lesser importance is given the reciprocal of that value. The redundancy of AHP does produce consistent requirements; however it also makes the process impractical for all but small projects [4], [5].

Several other methods employ a variation of the pair-wise comparisons performed for AHP. Hierarchy AHP is the most closely related; the process is nearly identical to AHP, except that requirements are first subjectively prioritized as low, medium, or high. Pair-wise comparisons are then performed on the requirements of each group [5]. Other algorithms, such as a binary search tree, and bubble-sort have also been used to compare requirements in pairs. With the exception of bubble sort, these methods require fewer comparisons than AHP. However they are still not feasible for larger projects, nor do they provide the same level of consistency [5].

Total Quality Management (TQM) and Quality Function Deployment (QFD) are two other quantitative methods used to prioritize requirements. TQM ranks requirements against a set of criteria that have been deemed necessary for the success of a project [4]. A priority rank is then determined based on the weight of the success criteria and the requirement. QFD correlates the value a proposed product feature has to a customer with specific requirements in order to determine priority. These methods are regarded as robust, however it is well known that the time and commitment needed to execute them has prevented their wide-scale adoption by organizations [4].

Despite the myriad of methods that have been proposed, research suggests that none have gained universal acclaim, nor have they been widely adopted [3]. While some methods like AHP, QFD, and TQM seem to produce more consistent prioritizations, [4], [5] they are also complex, time-consuming, and difficult to implement [4]. Other less formal methods may save time initially, but could cause problems in the later stages of a project if appropriate factors are not accounted for. Lehtola's study on the practical challenges of RPM suggests that project managers do not have access to a method that is both simple *and* effective [4]. In order to increase the effectiveness of requirements prioritization, new methods need to be developed that save time, yet preserve the accuracy that more robust methods currently offer.

III. Solution Approach

To properly evaluate the quality and priority of requirements in software projects, analysts must follow a methodology that takes into consideration the quality attributes of requirements that are considered important for specific software projects. In addition, the methodology must provide capabilities to determine the relative importance of each identified quality attribute. This would allow the methodology to provide a requirement prioritization scheme that represent how well requirements meet quality attributes and how important those quality attributes are for the identified software project.

To create such methodology, the following approach is proposed. First, once requirements are elicited, a set of quality attributes are identified as evaluation criteria. These attributes are defined in terms of many different features, where each feature is determined to be present or not. Once all features are identified, each requirement is evaluated against each feature using a simple binary scale (i.e., 0 or 1). Requirements that satisfy the highest number of features would expose a higher level of quality (or priority) for that particular quality attribute. Once all requirements are evaluated and measurements computed for all features, the proposed approach uses desirability functions to fuse all measurements into one unified value that is representative of the overall quality of the requirement. This unified value is computed by using a set of desirability functions that take into consideration the priority of each quality attribute. Therefore, the resulting priority of each requirement is derived from decision-makers' goals for a specific software project. This result in a requirement prioritization approach based on how well requirements meet quality attributes and how important those quality attributes are for the identified software project.

IV. Desirability Functions

Desirability functions are a popular approach for simultaneous optimization of multiple responses [6], [7]. They have been used extensively in the literature for process optimization in industrial settings, where finding a set of

operating conditions that optimize all responses for a particular system is desired. Through desirability functions, each system response y_i is converted into an individual function d_i that varies over the range $0 \leq d_i \leq 1$, where $d_i = 1$ when a goal is met, and $d_i = 0$ otherwise [7]. Once each response is transformed, the levels of each factor are typically chosen to maximize the overall desirability, which is represented as the geometric mean of all m transformed responses [6]. Alternatively, when factors are uncontrollable by analyst, the overall desirability value can be used to characterize the system based on the multiple selected criteria.

Similar to the characterization of industrial processes, the evaluation of the quality of requirements in software systems can be approached by finding the set of criteria that provide the optimal benefit vs. cost value for a particular application. When formulated this way, desirability functions can be used to provide a unified measurement that characterizes the quality of requirements based on a set of predefined project criteria. Once the desirability of all requirements is computed, analyst can use this information to determine the relative priority of requirements and select the best ones simply by choosing the most desirable one for a particular project.

A. Computing Desirability

The first step in the desirability functions approach involves the selection of requirements for a particular task. Ideally, the initial list of requirements would be easily identified for the specified assignment. However, in most practical scenarios this is not the case; leaving requirement analysts with the complex task of eliciting requirements from multiple sources, deriving requirements from one or more imposed requirement, and disambiguating the existing set. The results of these non-trivial efforts are captured in the requirements vector, as presented in (1).

$$X = \begin{bmatrix} n_1 \\ n_2 \\ \vdots \\ n_n \end{bmatrix} \tag{1}$$

Once the requirement vector is identified, each requirement can be evaluated against a set of quality attributes QA_1, QA_2,.., QA_n. The evaluation process takes places as follow. First, each quality attribute is defined in terms of m features, where $m>1$. The evaluation scale for each feature is binary; that is, the feature is evaluated as being present/true or missing/false. For example, requirements can be prioritized based on their type. In this case, the quality attribute *Type* can be defined with the following features: *Functional*, *Imposed*, and *Product*. Typically, a functional requirement imposed by the customer—as opposed to derived by the development team—on the product itself (instead of on the process) would be of higher priority. Therefore, the highest priority requirement (based on the *Type* quality attribute) would be one where *Functional=1*, *Imposed=1*, and *Product=1*. Similarly, the lowest priority requirement based on the Type quality attribute is one where *Functional=0*, *Imposed=0*, and *Product=0*. With this framework in place, a measurement of the importance of the j^{th} requirement based on the i^{th} quality attribute (e.g., *Type*) can be computed using (2),

$$y_{ij} = \frac{\sum_{x=0}^{m} f_x}{m} \tag{2}$$

where m is the number of features identified for the i^{th} quality attribute. This computation normalizes the evaluation criteria to a scale of 0 – 100, where 0 represents the lowest score and 100 the highest. The overall assessment of the requirement set based on all quality attributes is captured using the quality assessment matrix Q presented in (3). As seen, each y_{ij} value of the matrix represents the score of the j^{th} requirement based on each individual i^{th} quality attribute. It is important to point out that the quality assessment matrix can be extended to evaluate requirements based on any quality attributes containing numerous features.

$$Q = \begin{array}{c} \begin{matrix} QA_1 & QA_2 & \cdots & QA_m \end{matrix} \\ \begin{bmatrix} y_{11} & y_{21} & \cdots & y_{m1} \\ y_{12} & y_{22} & \cdots & y_{m2} \\ \vdots & \vdots & \ddots & \vdots \\ y_{1n} & y_{2n} & \cdots & y_{mn} \end{bmatrix} \end{array} \tag{3}$$

Finally, to assess the importance of each quality attribute, a weight vector W is created where r_i represents the importance of the QA_i quality attribute using the scale 0 – 10, where 0 represents lowest importance and 10 represents highest importance. The weight vector W is presented in (4).

$$W = \begin{bmatrix} r_1 \\ r_2 \\ \vdots \\ r_m \end{bmatrix} \tag{4}$$

Once the information of X, Q, and W is collected, desirability values for each requirement can be computed using the desirability matrix d presented in (5). As seen, each d_{ij} value of the matrix represents the desirability of the j^{th} requirement based on each individual i^{th} quality attribute.

$$d = \begin{bmatrix} d_{11} & d_{21} & \cdots & d_{m1} \\ d_{12} & d_{22} & \cdots & d_{m2} \\ \vdots & \vdots & \ddots & \vdots \\ d_{1n} & d_{2n} & \cdots & d_{mn} \end{bmatrix} \tag{5}$$

Each individual desirability value d_{ij} is computed according to requirement analysts' goals. For example, quality attributes that are represented positively by a higher y_{ij} value are transformed using the maximization function in (6) [7]. Alternatively, quality attributes that are represented negatively by a higher y_{ij} value (e.g., penalties such as cost and risk) are transformed using the minimization function in (7) [7],

$$d_{ij} = \begin{cases} 0 & y_{ij} \le L \\ \left(\dfrac{y_{ij} - L}{T - L} \right)^{r_i} & L \le y_{ij} \le T \\ 1 & y_{ij} > T \end{cases} \quad (6)$$

$$d_{ij} = \begin{cases} 1 & y_{ij} < T \\ \left(\dfrac{U - y_{ij}}{U - T} \right)^{r_i} & T \le y_{ij} \le U \\ 0 & y_{ij} > U \end{cases} \quad (7)$$

where L and U are the lower and upper limits, T is the target objective (e.g., 100 for maximization, 0 for minimization), and r_i is the desirability weight for the i^{th} quality attribute. It is important to note that (6) and (7) are the normal equations for the desirability function approach. However, through experimentation, it was found that the approach for requirements prioritization performed better when $d_{ij} > 0$. Therefore, as heuristic, when d_{ij} is less than .0001, the d_{ij} value is set to .0001. A desirability weight of $r = 1$ results in a linear desirability function; however, when $r > 1$, curvature is exposed by the desirability function to emphasize on being close to the target objective (T). When $0 < r < 1$, being close to the target objective is less important. Once individual desirability values for each quality attribute are computed, the overall requirement desirability value can be computed using (8). As seen, each overall desirability value is computed as the geometric mean of all m individual desirability values for requirements $1, 2, \ldots, n$.

$$D = \begin{bmatrix} \left(\prod_{i=1}^{m} d_{i1} \right)^{1/m} \\ \left(\prod_{i=1}^{m} d_{i2} \right)^{1/m} \\ \vdots \\ \left(\prod_{i=1}^{m} d_{in} \right)^{1/m} \end{bmatrix} \quad (8)$$

Once the overall desirability value is computed for all requirements, requirement analysts' can use this value as a priority measurement derived from the predefined quality attributes and their relative importance for the project.

V. Case Study

This section presents results of a requirement prioritization case study using the proposed approach. The case study evaluates 10 requirements based on the following identified quality attributes: Type, Scope, Customers Satisfaction, Perceived Impact (PMF), Application-Specific Attributes, and Penalties.

1) Type: The type of the requirement. Requirement type is defined with the following features: Functional, Imposed, and Product.

2) Scope: The scope of the requirement. This quality attribute asseses the impact of this requirmenent on the overall system. Requirements that affect many (or all) subsystems are determined to have higher priority than requirements that affect minimal number subsystems. Scope is defined with the following features: Subsystem 1 (S1), Subsystem 2 (S2), …, Subsystem n (Sn)

3) Customer Satisfaction:The number of customers the requirement satisfies. The higher the number of customer the requirement satisfies, the higher the desirabilty of the requirement. Customer Satisfaction is defined with the following features: Customer 1 (C1), Customer 2 (C2), …, Customer n (Cn)

4) Perceived Impact (PMF): The perceived impact the requirement has on the project based on expert opinion. This quality attribute asks each software lead the question "Is this requirement Perceived as a Major Functionality (PMF)?". Perceived Impact is defined in terms of all leads (software, hardware, systems). Therefore the features are: Lead 1 (L1), Lead 2 (L2), …, Lead n (Ln)

5) Application-Specific: The attributes that are important to the specific software application. Depending on the application domain (e.g., safety critical systems), requirements dictating a specific functionality will have higher importance. In this case study, application-specific is defined with the following features: Usability (U), Performance (P), Safety (S), Security (S), Reliability, and Interoperability (I).

6) Penalties: The penalties associated with the requirement. Requriements are associated with varied types of penalties, for example cost, risk, complexities, etc. This quality attribute is designed to ask the question "Is the requirement perceived as costly/risky/complex?". Penalties is defined with the following features: Costly (C), Risky (R), and Complex (Cx).

Using synthetic data for the identified quality attributes and the parameters presented in Table I, the binary input

evaluation, individual requirement desirability values, and overall requirement desirability (i.e., *D*) are presented in Tables II and III. As seen in Table I, all lower and upper boundaries are set to 0 and 100 respectively. Also, the quality attribute 3 (i.e., customer satisfaction) has been identified as having the highest priority. This is accomplished by setting the weight *r=5*, where as all other weights are set to *r=1*. Finally, the target values for quality attributes 2, 4, and 5 have been set to 70. This means that for QA2, QA4, and QA5, the requirements in (1) are considered 100% desirable if they meet or exceed 70% of each quality attribute's features.

TABLE I - DESIRABILITY FUNCTION PARAMETERS

Parameters	Benefits					Cost
	QA1	QA2	QA3	QA4	QA5	QA6
Lower (L)	0	0	0	0	0	0
Upper (U)	100	100	100	100	100	100
Target (T)	100	70	100	70	70	0
Weight *(r)*	1	1	5	1	1	1

As seen, each requirement has been evaluated using the identified features for each quality attribute. The binary input scale is used to determine the presence of features. Using the proposed approach, the most desirable requirement (based on the quality attributes) is R8, followed by R4, R5, and so on. It is important to notice the following. When evaluating R8 for QA1, the resulting individual desirability value is 100% because R8 is a functional requirement, imposed by the customer, and a product requirement. That is, all features of QA1 are present in R8. However, when evaluating R8 for QA5, since the target value (T) is 70%, there is more latitude to having missing features and still obtain a high individual desirability value. In this case, the resulting desirability value for QA5 is 95%. Similar to this case study, project-specific parameters can be specified for the desirability function to properly prioritize the requirements in industry scenarios.

TABLE II – BINARY INPUT EVALUATION

Req	QA1=Type			QA2=Scope			QA3=Customers				QA4=PMF				QA5=App-Specific						QA6=Penalty		
	Func	Imp	Prod	S1	S2	S2	C1	C2	C3	C4	L1	L2	L3	L4	U	P	S	SEC	R	I	C	R	Cx
R1	1	0	1	0	1	1	1	0	0	1	1	0	0	1	1	1	0	1	0	1	1	1	1
R2	1	1	0	0	1	0	1	1	1	0	1	1	0	1	0	1	0	1	1	1	1	1	0
R3	1	1	1	0	1	0	0	0	1	1	1	1	0	0	0	1	1	0	0	1	0	0	1
R4	1	1	0	1	1	1	1	1	0	1	1	1	1	0	0	1	0	1	0	0	0	0	1
R5	1	1	1	1	0	0	1	0	1	1	0	0	1	0	0	0	1	1	1	1	0	1	0
R6	0	1	1	1	1	1	0	1	0	0	0	1	0	1	0	0	0	0	1	1	1	0	0
R7	0	1	0	0	1	1	0	1	0	0	0	1	0	1	0	0	0	1	0	1	1	1	1
R8	1	1	1	1	0	1	1	1	0	1	1	1	1	0	1	1	1	0	1	0	0	0	1
R9	1	1	1	0	1	0	0	1	1	0	0	1	0	1	0	1	0	1	0	1	1	1	1
R10	0	0	1	1	1	1	1	1	1	0	1	1	1	1	0	0	1	1	0	0	1	1	0

TABLE III – QUALITY MATRIX IN TABULAR FORM

Req	QA1=Type			QA2=Scope			QA3=Customers				QA4=PMF				QA5=App-Specific						QA6=Penalty			Overall Desirability
	Func	Imp	Prod	L1	L2	L3	C1	C2	C3	C4	L1	L2	L3	L4	U	P	S	SEC	R	I	C	R	Cx	
R1	0.6667			0.9524			0.0313				0.7143				0.9524						0.0001			10.51%
R2	0.6667			0.4762			0.2373				1.0000				0.9524						0.3333			53.68%
R3	1.0000			0.4762			0.0313				0.7143				0.7143						0.6667			41.44%
R4	0.6667			1.0000			0.2373				1.0000				0.4762						0.6667			60.74%
R5	1.0000			0.4762			0.2373				0.3571				0.9524						0.6667			54.30%
R6	0.6667			1.0000			0.0010				0.7143				0.4762						0.6667			22.99%
R7	0.3333			0.9524			0.0010				0.7143				0.4762						0.0001			4.68%
R8	1.0000			0.9524			0.2373				1.0000				0.9524						0.6667			72.36%
R9	1.0000			0.4762			0.0313				0.7143				0.7143						0.0001			9.55%
R10	0.3333			1.0000			0.2373				1.0000				0.4762						0.3333			48.21%

VI. CONCLUSION

The research presented in this paper develops an innovative approach for evaluating the quality of requirements in software projects based on multiple quality evaluation criteria. Specifically, it presents a methodology that uses Desirability Functions to create a unified measurement that represents how well requirements meet quality attributes and how important the quality attributes are for the project. Through a case study, the approach is proven successful in providing a way for measuring the quality of requirements for specific projects.

There are several important contributions from this research. First, the approach is simple and readily available for implementation using a simple spreadsheet. This can promote usage in practical scenarios, where highly complex methodologies for requirement evaluation are impractical. Second, the approach fuses multiple evaluation criteria and features to provide a holistic view of the overall requirement quality. In addition, the approach is easily extended to include additional quality attributes not considered in this research. Finally, the approach provides a mechanism to evaluate the quality of requirements in various domains. By modifying the parameters of the desirability functions, quality and priority of requirements can be evaluated by taking consideration of prioritized quality attributes that are necessary for different software domains. Overall, the approach presented in this research proved to be a feasible technique for efficiently evaluating the quality and priority of requirements in software projects.

VII. REFERENCES

[1] Svensson, R., Gorscheck, T., Regnell, B., Torkar, R., Shahrokni, A., Feldt, R. (n.d.), Quality Requirements in Industrial Practice - an interview study at eleven case organizations [Online]. Available: http://richard.torkar.googlepages.com/QRinindustry_RBS.pdf Mar. 2010.

[2] Herrmann, A., Daneva, M., "Requirements Prioritization Based on Benefit and Cost Prediction: An Agenda for Future Research," 16th IEEE International Requirements Engineering Conference, 2008.

[3] Lehtola, L., Kauppinen, M., & Kujala, "Requirements Prioritization Challenges in Practice," Proceedings of 5th International Conference on Product Focused Software Process Improvement, pp. 497-508, 2004.

[4] Weigers, K. E., "First Things First: Prioritizing Requirements," *Software Development*, vol. 7, no. 9, 1999.

[5] Karlsson, J., Wohlin, C., Regnell, B., "An Evaluation of Methods for Prioritizing Software Requirements," *Information and Software Technology*, vol. 39, pp. 939-947, 1998.

[6] Derringer, G., Suich, R., "Simultaneous Optimization of Several Response Variables," *Journal of Quality Technology*, vol. 12, pp. 214-219, 1980.

[7] Montgomery, D., *Design and Analysis of Experiments*, Wiley, 7th Edition, 2008.