Available Online at www.ijcsmc.com

International Journal of Computer Science and Mobile Computing

A Monthly Journal of Computer Science and Information Technology

ISSN 2320–088X

IJCSMC, Vol. 4, Issue. 5, May 2015, pg.202 – 208

RESEARCH ARTICLE

Interactive Precedence of Test Cases in a Scenario Using Genetic Algorithm

Amit Kumar Vishwakarma, Mr. Mahtab Alam
Computer Science & Noida International University, India
HOD, Dept. of Computer Science & Noida International University
amitkr.vishu@outlook.com
alam12mahtab@gmail.com

Abstract— This paper presents a methodology for requirement prioritization. The approach provides a flexible and realistic approach that considers all attributes of particular requirements and fuses them into a unified metric, 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 methodology addresses flaws from previous work and provides mechanisms for improving its suitability for practical applications.

Keywords— Requirements Engineering, Requirements Prioritization, Desirability Functions, Software Engineering, Agile methodology, Genetic Algorithm

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 multitude of purposes, such as controlling safety-critical systems and playing a key role in national security, finances, entertainment, and educational systems. As the prevalence of software increases, so does the complexity, as well as the number of requirements that are derived for modern software projects.

One of the keys to making the right decision is to prioritize between different alternatives. must be taken into consideration. For example, when buying a new car, it is relatively It is often not obvious which choice is better, because several aspects easy to make a choice based on speed alone (one only needs to evaluate which car is the fastest). When considering multiple aspects, such as price, safety, comfort, or luggage load, the choice becomes much harder. When developing software systems, similar trade-offs must be made. The functionality that is most important for the customers might not be as important when other aspects (e.g. price) are factored in. We need to develop the functionality that is most desired by the customers, as well as least risky, least costly, and so forth. Prioritization helps to cope with these complex decision problems. This chapter provides a description of available techniques and methods, and how to approach a prioritization situation.

II. IMPLEMENTATION 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 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. Most requirements prioritization methods involve examining requirements through the framework of benefit and cost. 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. 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. One of the most consistent methods that have been developed is the GENETIC ALGORITHM. The genetic algorithm is a method for solving both constrained and unconstrained optimization problems that is based on natural selection, the process that drives biological evolution. The genetic algorithm repeatedly modifies a population of individual solutions. At each step, the genetic algorithm selects individuals at random from the current population to be parents and uses them to produce the children for the next generation. Over successive generations, the population "evolves" toward an optimal solution. You can apply the genetic algorithm to solve a variety of optimization problems that are not well suited for standard optimization algorithms, including problems in which the objective function is discontinuous, non differentiable, stochastic, or highly nonlinear. The genetic algorithm can address problems of *mixed integer programming*, where some components are restricted to be integer-valued.

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.

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 Fig ...1

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

Fig.1

Once the requirement vector is identified, each requirement can be evaluated against a set of quality attributes *QA1*, *QA2*,.., *QAn*. 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 *jth* requirement based on the *ith* quality attribute (e.g., *Type*) can be computed using
From Fig.2

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

Fig.2

where *m* is the number of features identified for the *ith* quality attribute. where *m* is the number of features identified for the *ith* 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 *yij* value of the matrix represents the score of the *jth* requirement based on each individual *ith* 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. Shown in Fig 3

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

Fig.3

Finally, to assess the importance of each quality attribute, a weight vector *W* is created where *ri* represents the importance of the *QAi* quality attribute using the scale 0 – 10, where 0 represents lowest importance and 10 represents highest importance. The weight vector *W* is presented in Fig.4

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

Fig.4

In addition, a lower and upper limit vectors, expressing lows and upper limits on desirability values, as well as a target vector expressing target desirability values, are created for all attributes as follows

$$L= \begin{bmatrix} L1\ L2..Lm \end{bmatrix}$$

$$U= \begin{bmatrix} U1\ U2..U\ m \end{bmatrix}$$

$$T = \begin{bmatrix} T1\ T2..Tm \end{bmatrix}$$

Once the information of *R*, *Q*, *W*, *L*, *U*, and *T* is collected, desirability of attribute *qj* for requirement *ri* can be computed as follows in Fig.5 and Fig.6

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

Fig.5

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

Fig.6

where L and U are the lower and upper limits, T is the target objective (e.g., 100 for maximization, 0 for minimization), and *ri* is the desirability weight for the *ith* quality attribute. It is important to note that (5) and (6) are the normal equations for the desirability function approach. However, through experimentation, it was found that the approach for requirements prioritization performed better when *dij > 0*. Therefore, as heuristic, when *dij* is less than .0001, the *dij* 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*, …, *n*. Shown in Fig.7

$$D = \begin{bmatrix} \left(\prod_{j=1}^{m} d_{1j}\right)^{1/m} \\ \left(\prod_{j=1}^{m} d_{2j}\right)^{1/m} \\ \vdots \\ \left(\prod_{j=1}^{m} d_{nj}\right)^{1/m} \end{bmatrix}$$

Fig.7

Once the overall desirability value is computed for all requirements, then goes to genetic algorithm, for crossover operator and mutation operator .

B. Crossover Operator

- Prime distinguished factor of GA from other optimization techniques
- Two individuals are chosen from the population using the selection operator
- A crossover site along the bit strings is randomly chosen
- The values of the two strings are exchanged up to this point
- If S1=000000 and s2=111111 and the crossover point is 2 then S1'=110000 and s2'=001111
- The two new offspring created from this mating are put into the next generation of the population
- By recombining portions of good individuals, this process is likely to create even better individuals.

C. Mutation Operator

- With some low probability, a portion of the new individuals will have some of their bits flipped.
- Its purpose is to maintain diversity within the population and inhibit premature convergence.
- Mutation alone induces a random walk through the search space
- Mutation and selection (without crossover) create a parallel, noise-tolerant, hill-climbing algorithms

IV. CASE STUDY

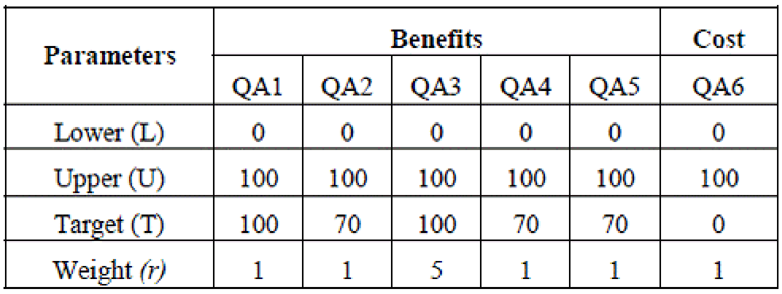
THIS SECTION PRESENTS RESULTS OF A REQUIREMENT PRIORITIZATION CASE STUDY USING THE PROPOSED APPROACH. THE CASE STUDY EVALUATES 10 REQUIREMENTS BASED ON THE FOLLOWING IDENTIFIED QUALITY ATTRIBUTES: TYPE, SCOPE, CUSTOMERS SATISFACTION, PERCEIVED IMPACT (PMF), APPLICATION-SPECIFIC ATTRIBUTES, AND PENALTIES.

*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 assess the impact of this requirement 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)

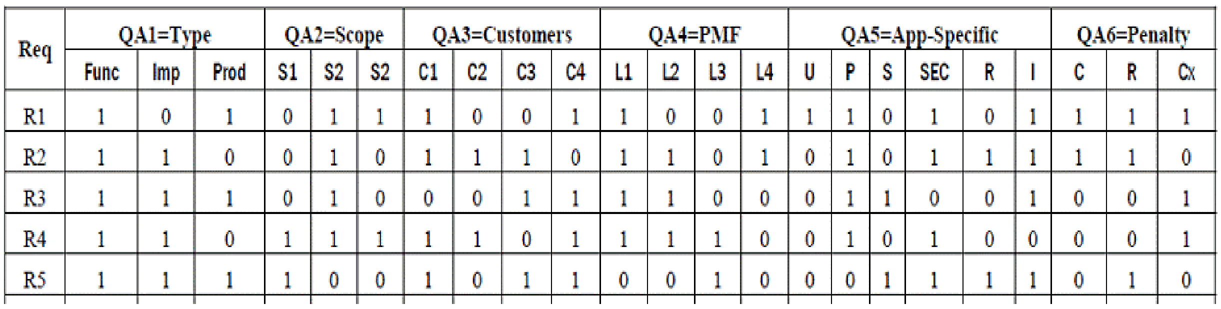
*5) **Application-Specific:** The attributes that are important to the specific software application. Depending on the application domain (e.g., safety critical systems), requirements dictating a specific functionality will have higher importance. In this case study, application-specific is defined with the following features: Usability (U), Performance (P), Safety (S), Security (S), Reliability, and Interoperability (I).*
6) ***Penalties:** The penalties associated with the requirement. Requirements 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)*
This section presents results of a requirement prioritization case study using the proposed approach. The case study evaluates 10 requirements based on the following suggested quality attributes: Type, Scope, Customers Satisfaction, Perceived Impact (PMF), Application-Specific Attributes, and Penalties [1a]. Using synthetic data for the identified quality attributes in Table II and the attributed parameters in Table I, the matrix *Q* and *d* are computed where the values of the latter are shown in Table III. Finally, the overall desirability vector *D* is compute based on the matrix *d*. This vector is shown in the Overall Desirability column in Fig.8



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

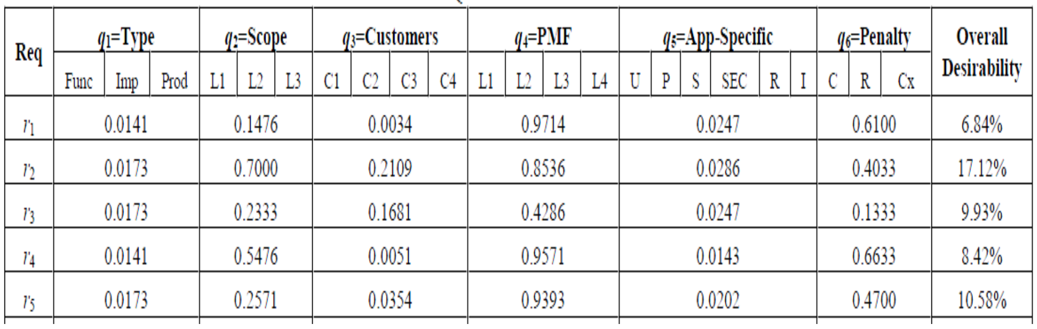
Fig.8

All lower and upper boundaries are set to0 and 100 respectively. Also, *Customer Satisfaction q*3 has been identified as having the highest priority. This is accomplished by setting the weight *w*3 = 5, where as all other weights are set to 1. Finally, the target values for quality attributes *q*2, *q*4, and *q*5 have been set to 70. This means that for *q*2, *q*4, and *q*5, the requirements in (1) are considered100% desirable if they meet or exceed 70% of each quality attribute's features. 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 for attributes with dissimilar features such as *q*1 while the nonbinary input scale is used to determine the importance of a feature in a given attribute for attributes with similar features such as *q*3. Using the proposed approach, the most desirable requirement (based on the quality attributes) is *r*2, followed by *r*10, *r*7, and so on. It is important to note that the desirability of *r*9 regarding the *Scope* attribute *q*2 is 96.19% when the target of this attribute *T*2 is only 70%. This shows that there is more latitude to having missing features and still obtaining a high individual desirability value. Shown in Fig.9and Fig.10



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

Fig.9



Req	q_1=Type			q_2=Scope			q_3=Customers				q_4=PMF				q_5=App-Specific						q_6=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	
r_1	0.0141			0.1476			0.0034				0.9714				0.0247						0.6100			6.84%
r_2	0.0173			0.7000			0.2109				0.8536				0.0286						0.4033			17.12%
r_3	0.0173			0.2333			0.1681				0.4286				0.0247						0.1333			9.93%
r_4	0.0141			0.5476			0.0051				0.9571				0.0143						0.6633			8.42%
r_5	0.0173			0.2571			0.0354				0.9393				0.0202						0.4700			10.58%

Fig.10

V. CONCLUSIONS

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.

REFERENCES

[1] J. Karlsson, C. Wohlin, B. Regnell, “An Evaluation of Methods for Prioritizing Software Requirements,” *Information and Software Technology*, vol. 39, no. 14-15, July 1998, pp. 939-947.

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

[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] P. Laurent, J. Cleland-Huang, C. Duan, “Towards Automated Requirements Triage,” *International Requirements Engineering Conference*, New Delhi, India, October 2007, pp. 131-140.

[5] P. Berander, *Evolving Prioritization for Software Product Management*, Ph.D. Dissertation, Department of Systems and Softwaer Engineering, Bleking Institute of Technology, Sweden, 2007.

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

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

[8] S. I. Mohamed, I. A. ElMaddah, A. M. Wahba, “Towards Value-Based Requirements Prioritization for Software Product Management,” *International Journal of Software Engineering*, vol. 1, no. 2. July 2008, pp. 35-48.

[9] 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.

[10] S. A. Marjaie, V. Kukarni, “Recognition of Hiddern Factors in Requirements Prioritization Using Factor Analysis,” *Interntional Conference on Computational Intelligence and Software Engineering*, Wuhan, China, December 2010, pp. 1-5.