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
IMPACT FACTOR: 6.017

IJCSMC, Vol. 8, Issue. 4, April 2019, pg.51 – 58

Prioritization of Test Cases Using Agile Methodology

Vatsala Arora[1]; Nimisha Rohatgi[2]; Sheetal Sharma[3]; Prakash Srivastava[4]

Department of Computer Science and Engineering, Amity University, India
[1] aroravatsala09@gmail.com; [2] nimisharohatgi@gmail.com; [3] ssharma31@amity.edu; [4] psrivastava9@amity.edu

Abstract— Software management has a very important play in the advancement and expansion of various other industries. The reason is the use of machines to automate the tasks which were earlier performed manually. The prime intention of every organization is to maximize their profits. One significant way to do this is to invest on improvising the life cycle of its processes. After the software undergoes production, it should result in a greater output. Therefore, it is critical to do this using an agile methodology so that the manual as well as the automated processes can be done side by side with the perfect balance. Automatically, the input from the tester in the form of test case specification becomes of utmost importance. Throughout the process the customer becomes a part of the development team to provide inputs as well as give feedback according to its prioritization. Agile methodology has been proposed to keep the balance between the customer test cases and the development of the software.

Keywords— test case, prioritization, agile, desirability factor, mutation; crossover, disagreement factor

I. INTRODUCTION

This Agile methodology is extensively used to prioritize the test cases of the software. The prioritization process is an addition to the software testing process.[1] The priority of the test cases is based on the test cases of the customer from the software. Through this process, the developer will be clear about the expectation of the customer from the development team. It will increase the quality of the software which will be delivered to the client.[2]

It is mostly preferred by the developers that the software they develop is of high quality. The most practical approach in the development of high value software is to rank the test cases in order of the priority. The similar procedure goes for test cases. By giving the input of a test plan, we can select the important test cases and further rank them by prioritizing them based on their quality attributes. Since the test cases keep changing, we need a stable and flexible methodology. However, if we study the methods of prioritization which are currently available, they are often complex and very challenging to implement. Most of this times, this becomes a problem for the clients as they need to have the proper knowledge and expertise to use these software. The approach we use in this paper assesses the attributes of the test case. The quality attribute measurement is easy to calculate to determine the priority of the test case or the test case.[3]-[5] Furthermore, since we are using agile methodology as our approach, it is important for our software to be interactive. So, we have used Interactive Genetic Algorithm to produce an ordering of the test cases and to take the preference of the customer into consideration. After this, we will calculate the disagreement factor to satisfy the functional constraints of the software which is done by forming pair among the test cases by the system and the client.

Considering the perspective of the agile methodology, we take the test cases of the user as the input and give the test cases after prioritization as the best output. This happens as a result of the evaluation by the system as well as the customer. In this software, the client may input his test cases in the form of a paragraph as a test plan or through graphical representation by using use case diagrams. The use can diagrams can be simply drawn through drag and drop options which will be inbuilt in the software. After the specification of the test cases, we parse them into some meaningful test cases. Post to this step, we will apply some quality attributes as well as some sub-attributes. In this paper, we have taken six parent attributes and eighteen child attributes as the base. We have assumed that these attributes are manually applied by the project manager of the organization operating these test cases.[6]

The system will then calculate a value known as the Desirability Factor. This desirability factor will basically determine the importance of the test cases which were previously parsed. If the value of the desirability factor is high, this means that the test case or the test case is desired to be kept at a lower priority. Therefore, the priority and the desirability factor of a test case are inversely proportional to each other. If the desirability factor is ranked in the order of ascendance, then it will give us the test cases aligned according to its priority. Also, it should be noted that this ranking is only on the basis of calculations by the system. Once the software gives the prioritization, we input the test cases prioritization alignment by the clients. When we have both these prioritization orders, we input them to calculate a Disagreement Factor. This is done by forming pairs of the test cases and observing the deviation between the system and the user ranking. After this, we apply the recursive genetic algorithm for minimization of the disagreement factor. We need the disagreement factor to be reduced to 0 or 1 so that maximum agreement can be reached by the system and the client. This disagreement factor is reduced by applying Mutation and Crossover on the pairs and then calculating the disagreement factor again. After applying all these procedure, what we get in the end is the best ranking of the test cases in the order of prioritization.

We can confidently claim that this application will help an organization to obtain test case prioritization by taking the customer inputs into consideration to give the best set out output ranking. One of the prime reasons which are accountable for the failure of a project is the incorrect analysis of the test cases of the customer. Parameters like time, cost, effort and human competencies should be while the development of a project. This project does intelligent processing and also takes customer's opinion as the higher is the involvement of a client in the analysis phase, the higher will be the chances of a project being a success.

II. SYSTEM DESCRIPTION

This project belongs to the software engineering domain where the different phases of the Software Development Life Cycle are implemented. The objective of our application is to gather the test plan, then select the test cases from the test plan. These can be elicited in the following two ways: first is though test-based story board form, and the second is through a use case diagram though a user controlled toolbox. The main objective of taking the input in two forms is to make the system more user-friendly.[7] The test cases are then evaluated on the basis of the attributes and the sub-attributes which are used to calculate the desirability factor. This desirability factor is value which tells us the priority of the test case. The desirability factor is used to calculate the overall desirability percentage based on which we rank the test cases. Lesser the desirability percentage, greater is the priority of the test case.

This application is implemented as console-based where we have assumed that the customer and the developer are virtually the same entities. They can also be located geographically at different locations around the world. The customer can easily provide the test cases to the developer in the form of text or graphics. The valid test cases are passed to the system through parsing to compute the desirability factor and to minimize the disagreement factor though an iterative and recursive procedure.

This application has the following functions:

1). Parsing: By the use of this functionality, the developer can easily select the meaningful test cases and eliminate those which are not required by the system. This takes place by tokenization where a comma or a full-stop is used for the filtration the test plan into test cases. The meaningful test cases can be passed as the input for the prioritization process. These are then stored in the database for further processing.

2).Desirability Factor: The desirability factor will determine the importance of the test case though a function. This calculation is mathematical and is done by the use of some pre-defined values. After the desirability factor is calculated, we calculate cumulative desirability percentage and the lesser the percentage, more is the priority.

3). Disagreement Factor: After the ranking is taken as an input from the user, the disagreement factor is calculated by forming pairs and then looking for dissimilar pairs. The count of the dissimilar pairs is the disagreement factor.

4). Mutation and Crossover: As the input of the user is also taken into consideration, it is important that the disagreement factor is below 2. However, if such is not the case, then the process of mutation and crossover from the genetic algorithm are used and

the disagreement factor is calculated again. If the disagreement factor is still not below 2, this process will go on recursively until the desirability factor reduces to 0 or 1. The mutation technique deals with the reversal of any test case pair like reversing the pair of the test cases. In crossover operation, we fix a split up point and thereafter exchange the pairs between them.

In the performance test cases, we ensure that the tokenization process is as efficient as possible since parsing is done on some logical basis. If the parsing process is incorrect, then the whole application process will be incorrect. It is also required that the filtration of test cases is done by an experienced Project Manager who is aware of the expected test cases of the organization. The valid test cases should be effectively stored in databases and the rest of them should be discarded. This is done so that there is no ambiguity or non-clarity while the further processing continues.

For the logical database test cases, we need a back-end which is compatible to the application to handle the heavy computational data. It is important to maintain the integrity of the database. This is due to the reason that the data is primarily stored and fetched from the database.[8]-[10] If the database is incompatible, then this whole process will be falsified. Therefore, we plan to use SQL Server Database that is efficient in terms of computational transactions.

The design constraints in our project need the developer to be able to furnish the test cases in the form of text as well as in the graphical form. We have given convenience to the customer in the form of user defined controls. It is also important that the extracted test cases of the previous projects do not interfere with the current database. Once we select all the attributes for the test cases, we write the code in such a fashion that the calculations find the most appropriate ranking or prioritization.

III. PROPOSED METHODOLOGY

In order to evaluate the priority of the test cases, there should be a methodology that takes quality attributes into consideration. Furthermore, the methodology should have the capability to determine the relative importance of the quality attributes with respect to each other.

Therefore, to create such a methodology, the following approach has been proposed. Firstly, after the test cases are inputted, the quality attributes become the evaluation criteria for these inputs. Each of the input is evaluated against each of these evaluation criteria. The test case that will satisfy the highest amount of evaluation criteria will become the most prioritized test case. This is done by computing the desirability function which is a unified value. Hence, the final result ranking based on priority has been derived from the decision maker's goals for a certain project. This approach is known as the solution approach where the result is based on how well the test cases meet up with the quality attributes and how important these attributes are for the particular software project.

Desirability functions, by far, have been quite a popular approach for the simultaneous optimization of multiple responses. There are used to find a set of conditions to optimize the responses for a specific system. The system response, y_i is converted into d_i which is an individual function, where $0<=d_i<=1$. The value of d_i will be 1 when the goal is met and the value of d_i will be 0 when the goal is not met. After the response has been transformed, in order to maximize the overall desirability, the level of each of the factor is chosen. This is represented though the geometric mean. The geometric mean is taken of all the transformed responses which are equal to *m*. Otherwise, if the factors are uncontrollable, the overall desirability value can characterize the system based on the many selected criteria.
The computation of the quality of the test cases can be approached by finding the set of criteria which will provide the prime benefits with the minimum cost value. Here, desirability functions may provide a unified measurement based on pre-defined criteria.

To calculate the desirability functions, the very first step includes selecting the test cases for a specified project. The vector *X* will store the result of the first step as follows,

$$X = \begin{bmatrix} n_1 \\ n_2 \\ \vdots \\ n_n \end{bmatrix} \qquad (1)$$

After the identification of this vector, the quality attributes are identified as QA_1, QA_2, ...QA_n. Following is how the evaluation process takes place. First of all, each of the *QA* is defined for *m* features ($m>1$). For each of the feature, the evaluation scale is in

binary. This means that of the attribute is to assessed for a particular test case then the binary value will be 1, otherwise it will be 0.

After the setup of this framework, y_{ij} is the measure of importance of the test case which will be computed for the j_{th} test case with the i_{th} quality as follows,

$$y_{ij} = \frac{\sum_{x=0}^{m} f_x}{m} \qquad (2)$$

In the above equation, m is the total number of features for the i_{th} quality. The value of y_{ij} will range from 0 to 100 where 0 is the lowest score and 100 is the highest score. Now, for each of the quality attributes, the relative importance is calculates and stored in an assessment matrix which is Q.

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

The last step in the calculation of the desirability function is to find the weight vector W. For the calculation of the weight vector, r_i will represent the importance of the quality attribute QA_i in the scale 0 to 10 where 0 is the lowest score and 10 is the highest score which is the score for importance. W, which is the weight vector is represented as follows,

$$W = \begin{bmatrix} r_1 \\ r_2 \\ \vdots \\ r_m \end{bmatrix} \qquad (4)$$

After collecting all the information in the form of vectors X, Q and W, the desirability matrix d is calculated for each of the test case The d_{ij} value of the matrix will represent the desirability of the j_{th} test case for the i_{th} quality attribute.

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

The desirability value d_{ij} is calculated based on the goals of the analysts. Higher y_{ij} value will be parallel to a positive quality attribute. On the other hand, lower value of y_{ij} is parallel to a negative quality attribute. The maximization function will represent the higher y_{ij} value as follows,

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

Whereas the minimization function is used to represent the lower values of y_{ij} like penalty, cost and risk,

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

In the equations (6) and (7), L is the lower limit value and U is the upper limit value. For the i_{th} quality attribute, the desirability weight is r_i.

It can be noted that the above equations (6) and (7) are normal equation in this desirability function approach that we are using. It was proved by experimentation that the best performance for the approach of test case prioritization is when $d_{ij}>0$. So, if the value of $d_{ij}<0.0001$ then it is set to 0.0001. The value of the desirability weight r =1 evaluates for a linear desirability function while r>1 results in a curvature which is close to T (Target Objective). After the calculation of the desirability value is successful for each individual quality, the overall desirability value is calculates as follow,

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

This value D can be used as the measurement of priority by the analysts in the final ranking of the test cases. This value is very useful to determine the importance of the test case relative to all the other test cases.

In this paper, we have used the following quality attributes as the evaluation criteria of the test cases,

1). Type: This attribute tells the type of the test cases. The sub attributes under this quality attributes are Functional, Imposed and Product.

2). Scope: This attribute tells the scope of the test case which is the impact of this on the system. If the test case will affect many subsystems then it will have a higher priority and if the test case affects a lesser number of subsystems then it will have a lesser scope. So we have the sub-attributes S1, S2, S3,.. Sn to determine the scope.

3). Customer Satisfaction: This attribute will assess the satisfaction level of the test case. The test case is tested against the sub-attributes which the customers C1, C2, ….., Cn to check the number of customers this test case can satisfy. If it satisfies a huge amount of customers then it will be highly desirable as compared to if it satisfies only quite a few customers.

4). Perceived Impact: This attribute is solely based on expert opinion. In this attribute, each of the project manager is asked if this test case is Perceived as a Major Functionality (PMF) or not. Since there can be a number of software leads, the sub attributes are L1, L2, …., Ln which will give their expert opinion on each of the test case.

5). Application Specific: Based on the application domain that is being used, it is determined whether this test case is specific to that application domain or not. If it has the specific functionality required, then it is highly desirable, otherwise it is not. Now the sub-attributes under this category are Usability (U), Interoperability (I), Performance (P), Security (Sec) and Safety (S).

6). Penalty: Complexities like cost, time and effort are substituted as penalties in this application. This attributes tells if the test case is costly or it will be time taking or it requires a lot of effort to be implemented. The sub attributes associated with this attribute are Complex (C_x), Costly (C) and Risky (R) which will be a negative impact on the test case.
Table I represents the various desirability function parameters for all the parent quality attributes,

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 it can be seen in Table I, the lower limit values, that is, L, is set as the lowest score which is 0 for all the six parent quality attributes. On the other hand, the upper limit, that is, U is set as the highest score which is 100. The target value, T varies for the quality attributes based on their target. The Target value is the highest score which is 100 for QA1 and QA3. The same value is 70 for QA2, QA4 and QA5 while it is 0 for QA6. This is because the cost should be minimum and the minimum value out of the range is 0. So the value of the Penalty or specifically the cost has been set as target value of 0. QA2, QA4 and QA5 have been considered 100% desirable while the quality attributes QA1 and QA3 are overall 70% desirable.

Now, the binary input scale will be used to see if these features or these attributes and sub-attributes are present in each of the test cases or test cases or not. For each of the quality attributes and the sub-attributes if the value of the binary input is 0 then the feature is present while if the value is 0 then the feature is absent from the table. The Table II shows the binary input evaluation of the quality attributes and the sub-attributes.

The example table we have used here, that is, Table II, has taken the binary input as 1 for Functional, Production, S2, S3, C1, C4, L1, L4, Usability, Security, Performance, Interoperability, Cost, Risk and Complexity for the test case with the test case id R1. For the same test case, it has taken, the binary input as 0 for Imposed, S1, C2, C3, L2, L3 and Safety which means that all the mentioned sub-attributes are absent from the test case.

Based on all of these inputs in the binary input evaluation table, that is Table 2, we calculate the desirability function for each of these test cases. The vectors, *X*, *Q* and *W* have been formed which are then used to form the desirability function matrix *d*. After that we calculate the overall desirability function for each of the test case through the equations (6), (7) and (8). In the given example, we see in Table III that while evaluating R8 for QA1, the desirability value is 1.000 which is equal to 100% which means that R8 is desirable for the corresponding quality attribute. Also, while evaluating R8 for QA5, the desirability equates to 95% since the target value for QA5 is 70%. Just like this example, many parameters can be specified for different applications.

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

IV. RESULTS AND DISCUSSIONS

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%

The Table III shows the quality matrix in a tabular form. Each row is for a test case which has the test case id R1 and each column has the attributes and the sub-attributes lined up. From here we can see that the final column is for the overall desirability which is calculated though the equation (8). These IDs can be ranked in the ascending order of their percentage. Therefore, in this example, the test case ID R7 will have the highest priority according to the system and the test case ID I R4 will have the least of the priority.

However, the processing does not end here. This is the prioritization given by the system. Thus processing uses agile methodology, therefore, the system will input the prioritization ranking by the customer. Thereafter, the disagreement factor is calculated. Interactive approach is used to prioritize the test cases in to give the final ranking.

V. CONCLUSIONS

This tool has been developed using a Genetic Algorithm conceptualization which is interactive in nature and helps an organization to build a product in an agile manner for value added business output. It helps software organizations to effectively prioritize the test cases by taking the customer's input as well. It will effectively calculate the desirability values based on experimentation performed. It will also calculate disagreement values iteratively and minimize the disagreement factor using Mutation and Crossover so that the user is able to obtain the best set of prioritized test cases in correspondence to the relevance of production level business output. This tool will not only help a software organization to identify the test cases clearly but also form a perspective of handling them in agile manner. It will help furnishing the test cases in the most convenient and appropriate way possible, that is, through text-based and graphics-based input. The main motive behind two levels of input is to provide convenience to the user so that they can furnish the test cases in every way possible. It will also help the organization to split and extract the test cases from a specified set of test cases easily by parsing it from points and commas conveniently segregating the input. It will help the organization to finally extract the best set of prioritized test cases or test cases being deciphered through an interactive way of iterative minimization which will enable them to understand the workflow of a software clearly and develop it efficiently.

ACKNOWLEDGEMENT

This term paper came to us as an excellent opportunity and gave us such an exposure. We would like to thank the authorities of **AMITY UNIVERSITY** for giving us this opportunity to work in this esteemed organization under the guidance of highly knowledgeable and experienced faculty Ms. Sheetal Sharma and Dr. Prakash Srivastava. We would also like to thank our DG ASET **Dr. Gurinder Singh** and our head of the Department **Dr. Abhay Bansal**, for their ongoing support throughout the project.

REFERENCES

[1] Svensson, R., Gorscheck, T., Regnell, B., Torkar, R., Shahrokni, A., Feldt, R. (n.d.), Quality Test cases 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., "Test cases Prioritization Based on Benefit and Cost Prediction: An Agenda for Future Research," 16th IEEE International Test cases Engineering Conference, 2008.

[3] Lehtola, L., Kauppinen, M., & Kujala, "Test cases 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 Test cases," *Software Development*, vol. 7, no. 9, 1999.
[5] Karlsson, J., Wohlin, C., Regnell, B., "An Evaluation of Methods for Prioritizing Software Test cases," *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.
[8] J. Karlsson and K. Ryan, "A cost-value approach for prioritizing test cases," Software IEEE, vol. 14, no. 5, pp. 67–74, 1997.
[9] J. Karlsson, "Software Test cases Prioritizing," in Proceedings of 2nd International Conference on Test cases Engineering (ICRE '96) , April 1996, pp. 110–116.
[10] S. Sivzittian and B. Nuseibeh, "Linking the Selection of Test cases to Market Value: A Portfolio - Based Approach," in REFSQ 2001, 2001.
[11] H. P. In, D. Olson, and T. Rodgers, "Multi-criteria preference analysis for systematic test cases negotiation," in COMPSAC 2002, 2002, pp. 887–892.
[12] F. Moisiadis, "Prioritising software test cases," in SERP 2002, June 2002.
[13] T. L. Saaty and L. G. Vargas, Models, Methods, Concepts & Applications of the Analytic Hierarchy Process. Kluwer Academic, 2000.
[14] D. Leffingwell and D. Widrig, Managing Software Test cases: A Unified Approach. Addison-Wesley Longman Inc., 2000.
[15] K. E. Wiegers, Software Test cases. Best Practices. Microsoft Press, 1999.
[16] S. Lauesen, Software test cases: styles and techniques. Addison Wesley, 2002.
[17] P. Avesani, C. Bazzanella, A. Perini, and A. Susi, "Facing scalability issues in test cases prioritization with machine learning techniques," in RE 2005, 2005, pp. 297–306.

