Volume 6, Issue 2, February 2016 ISSN: 2277 128X

International Journal of Advanced Research in Computer Science and Software Engineering

Research Paper

Available online at: www.ijarcsse.com

Participative Precedence of Software Requirements Using Recursive Genetic Algorithm Techniques

Neha Sharma
M.TECH (Research scholar)
Department of Computer Science & Engg.,
Krishna Engineering College,
Ghaziabad, Uttar Pradesh, India

Arun Aggarwal
Assistant Professor
Department of Computer Science & Engg.,
Krishna Engineering College,
Ghaziabad, Uttar Pradesh, India

Abstract-Requirements prioritization is an essential mechanism of agile software development approach. It maximizes the value delivered to the clients and accommodates changing requirements. The order in which requirements are implemented in a system affects the product delivered to the final users in the successive releases of the system.This leads to the formulation of software requirements, wherein a company decides the processes to be automated with specific functions.Once the requirements are formed, it becomes essential that they are developed on time with highest accuracy and speedy output.Therefore, it is important that the formulated requirements are developed in an agile manner with prioritization of the development of output-oriented requirements first so that manual processes can work in conjunction with the automated processes. This 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. Requirement analysts possess relevant knowledge about the relative importance of requirements. After prioritizing requirements according to quality based approach, we will use an Interactive Genetic Algorithm to produce a requirement ordering which complies with the existing priorities, satisfies the technical constraints and takes into account the relative preferences elicited from the user.

Keywords-requirements prioritization; interactive genetic algorithm; quality-based software engineering.

I. INTRODUCTION

Software industry plays a significant role in the growth of other industry verticals. Every industry has specific processes, methodologies and operations by the mode of which it fulfils its objectives.Every company wants to maximize its profit through increasing the returns on investments spent on such processes. These investments are in various forms such as Time, Effort and Money.In order to achieve this objective, companies consciously make an attempt to automate their life cycle processes in different departments so that intelligence, efficiency, integrity and accuracy could be brought into its culture.This leads to the formulation of software requirements wherein a company decides the processes to be automated with specific functions. Once the requirements are formed, it becomes essential that they are developed on time with highest accuracy and speedy output. . Therefore, it is important that the formulated requirements are developed in an agile manner with prioritization of the development of output-oriented requirements first so that manual processes can work in conjunction with the automated processes.

The precedence of software requirements is a critical part of the analysis phase because this prioritization needs to have a base and acceptance. In other words, if the customer approaches an IT vendor for the development of a software solution, it becomes essential for the vendor that it gives its part of requirements prioritization and also takes the expected prioritization of the customer in order to land on a win – win analysis of requirements.We aim to develop a system that takes user's software requirements as an input and outputs the best set of prioritized requirements as per system and customer evaluation. In this system, the user would be able to furnish requirements in a textual and graphical form. After the requirements have been specified, we shall parse them in order to determine the meaningful requirements. After parsing, we shall apply quality attributes and sub-attributes to the requirements. The system will calculate a numeric factor named as **Desirability Factor**. This factor shall determine how important any parsed requirement is. That is, lowest the desirability factor, higher the precedence of that requirement. Once the system has given its prioritization, we shall obtain the expected set of prioritized requirements from the customer and calculate a factor called **Disagreement Factor** by forming pairs. The differential pairs shall be the count of that factor. After the disagreement factor is yielded, we shall apply **recursive genetic algorithm** in order to minimize the disagreement count.The processes appliedunder thisalgorithm would be **Mutation & Crossover.**After the application of such processes, the final result set that we would obtain as the output shall be the best set of prioritized requirements as per system and customer.

II. LITERATURE REVIEW

2.1 The Sources for formulation of problem statement:

- A Quality-Based Requirement Prioritization Framework Using Binary Inputs (2010 IEEE).

- Using Interactive GA for Requirements Prioritization (2010 IEEE).
- A New Requirement Prioritization Model for Market Driven Products Using Analytical Hierarchical Process **(**2010 International Conference on Data Storage and Data Engineering).
- Reduction in number of comparisons for requirement prioritization using B- Tree. (2009 IEEE International Advance Computing Conference).
- A Broad, Quantitative Model for Making Early Requirements Decisions.(2008 IEEE)
- An Approach for Requirement Prioritization using B-Tree (2008 IEEE).
- Handling Uncertainty in Agile Requirement Prioritization and Scheduling Using Statistical Simulation**. (**Agile 2008 Conference).
- Requirements Prioritization with Respect to Geographically Distributed Stakeholders (2011 IEEE).
- An Algorithm for Priority Ranking of IndividualizedFunctional Requirements in Networked Software(2008 IEEE).

2.2 Summary of Relevant Papers:

Table1 - Summary of research papers:

Title of paper	A Quality-Based Requirement Prioritization Framework Using Binary Inputs	Requirements Prioritization with Respect to Geographically Distributed Stakeholders
Authors	Carlos E. Otero, Erica Dell, Abrar Qureshi(University of Virginia), Luis D. Otero(Florida Institute of Technology)	Azeem Ahmad, Aamir Shahzad, V. Kumar Padmanabhuni, Ali Mansoor, Sushma Joseph , Zaki Arshad(Blekinge Institute of Technology, Karl, Sweden)
Year Of Publication	2010 IEEE	2011 IEEE
Publishing Details	IEEE Computer Society, USA	IEEE Computer Society, Sweden
Summary	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. By modifying the parameters of the desirability functions,quality and priority of requirements can be evaluated. 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.	This paper discusses attributes that can affect the requirements prioritization when dealing with Geographically Distributed Stakeholders. People make their contribution to SDLC either by staying at one place or by being geographically distributed. The term GDS refers to those stakeholders, which are linked to a project and distributed all over the world. These distributions can be within same region or in a continent. These distributions can also be all over the world where each stakeholder has its own perception and expectations of the requirements in a software project This paper provides a framework that can beused to identify those requirements that can play animportant role in a product success during distributeddevelopment.
Web link	www.ieee.org/05489624	www.ieee.org/05952853

Continued…

Title of paper	Analytic Hierarchy Process (AHP), Weighted Scoring Method (WSM), and Hybrid Knowledge Based System (HKBS) for Software Selection: A Comparative Study	Using Interactive GA for Requirements Prioritization
Authors	Anil Jadhav, Rajendra Sonar (Indian Institute of Technology Bombay, India)	Paolo Tonella, Angelo Susi Fondazione Bruno Kessler(Software Engineering Research Unit Trento, Italy), Francis Palma(University of Trento)
Year Of Publication	2009	2010 IEEE
Publishing Details	ICETET-09	IEEE Computer Society, Italy

Summary	Analytical hierarchy process (AHP) and weighted scoring method (WSM) have widely been used for evaluation and selection of the software packages. This paper studies and compares these approaches by applying for evaluation and selection of the software components. The comparison shows that HKBS approach for evaluation and selection of the software packages is comparatively better than AHP and WSM with regard to (i) computational efficiency (ii) flexibility in problem solving (iii) knowledge reuse and (iv) consistency and presentation of the evaluation results.	This paper proposed an interactive genetic algorithm to collect pairwise information useful to prioritize the requirements for a software system. They have applied algorithm to a real case study, consisting of a non- trivial number of requirements, which makes AHP hardly applicable. This approach scaled to the size of the considered case study and produced a result that outperforms GA (i.e., a genetic algorithm which optimizes satisfaction of the initial constraints, without gathering further constraints from the user). Specifically, by eliciting between 50 and 100 pairwise comparisons from the user it was possible to obtain a substantially better ordering of the prioritized requirements.
Web link	www.ieee.org/05395484	www.ieee.org/05635176

2.3 Summary of Empirical Study:

A case study using the quality based approach helps a lot in understanding this approach. As in [QBA 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**. Using synthetic data for the identified quality attributes, the binary input evaluation, individual requirement desirability values, and overall requirement desirability was determined. All lower and upper boundaries were set to 0 and 100 respectively. Also, the quality attribute 3 (i.e., customer satisfaction) has been identified as having the highest priority. This was accomplished by setting the weight r=5, whereas all other weights are set to r=1.

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. Similar to this case study, project specific parameters can be specified for the desirability function to properly prioritize the requirements in industry scenarios.

The IGA algorithm was applied to prioritize the requirements for a real software system, as part of the project **ACube (Ambient Aware Assistance**). ACube is a large research project funded by the local government of the Autonomous Province of Trento, in Italy, aiming at designing a highly technological smart environment to be deployed in nursing homes to support medical and assistance staff.

As a product of the user requirements analysis phase, 60 user requirements (49 technical requirements) and three macro-scenarios were identified.

Specifically, the three macro scenarios are: (i) "localization and tracking to detect falls of patients", (ii) "localization and tracking to detect patients escaping from the nursing home", (iii) "identification of dangerous behaviors of patients"; plus (iv) a comprehensive scenario that involves the simultaneous presence of the previous three scenarios.

Out of these macro-scenarios, detailed scenarios have been analyzed together with the 49 technical requirements.

Together with the set of technical requirements, two sets of technical constraints have been collected during requirements elicitation: Priority and Dependency, representing respectively the priorities among requirements and their dependencies. Finally, for each of the four macro-scenarios, we obtained the Gold Standard (GS) prioritization from the software architect of the ACube project.

There are many research papers we have taken many ideas basically work on the requirement prioritization with the help of genetic algorithms.

III. SOLUTION APPROACH

The purpose of our application is to effectively gather the software requirements from the customer and prioritize them. The requirements could be elicited in two ways i.e. simple text-based story-board form and graphical form where the user can exhibit the requirements by drawing use case diagrams on a customized graphical editor through user control toolbox. After the proposed software requirements are gathered from the customer, they are evaluated on the basis of some attributes and sub-attributes in order to calculate the desirability values on the basis of which, they are prioritized. After prioritization, we take the user's input in relation to their expected prioritization and try to compare the prioritization of requirements from the developer and customer's perspective. The overall objective of the application is to minimize the disagreements that arise from the difference in prioritization of requirements.

Specific requirements

a) External interfaces: We implement this application as console-based wherein we assume that the developer and customer are virtually the same entities. In other words, the customer furnishes requirements to the developer and the

developer inputs them to this tool through text-based or graphical editor in order to extract the valid requirements, compute their desirability and minimize the disagreement. As a future scope, if we put this application on a private cloud, we assume that the customers will be able to access it on web.

b) Functions: The application has below functions

Input of requirements through a text-based or graphical editor: In this functionality, the developer could type the requirements in a text editor OR can input them through a text file. In the graphical editor, the developer can draw the requirements through a graphicseditor containing user-defined graphical use-case controls.

Extraction and storage of valid requirements: Through this functionality, the developer would be able to select the genuine requirements by checking them. The extraction of text-based requirements will occur through splitting and tokenization of requirements where a comma (,) or a full-stop (.) occurs in the text-based sentence input whereas in the graphical input, this extraction occurs on the basis of pixel distance from actor symbol to a function symbol, which is an oval.

Once the requirements are extracted, they are verified by the developer for their validity and then are stored in the database for use in next calculation.Then we will take the developer input about the sub-attributes that apply to the individual requirements through Yes or No.

The parent attributes and their sub-attributes will be fixed as below.

Table 2 – Input for extraction and storage of valid requirements:

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

Once the (Yes/No) input of individual attributes for individual requirements has been taken, a matrix as below will be generated, where 1 will signify a Yes and 0 will signify a No.

Table 3 - Output Matrix for extraction and storage of valid requirements:

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	0	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

Then, we will calculate the overall desirability factors of requirements with respect to the control matrix and formulae given below.

Table 4 – Matrix for overall desirability factors of requirements:

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

The desirability values of **first 5 attributes, per requirement** will be calculated based on the below formula.

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

The desirability values of the last attribute (Penalty), per requirement will be calculated based on the below formula.

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

[QBA Approach]

Table 5 – Quality Matrix in Tabular Form:

Req	QA1-Type			QA2-Scope			QA3-Customers				QA4-PMF				QA5-App-Specific						QA6-Pear			Overall Desirability
	Func	Imp	Prod	S1	S2	S2	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			0.3213				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				0.3572				0.9524						0.6667			50.30%

The requirements will be prioritized in the increasing order of desirability values. This prioritization will be of **developer's perspective**. Then the prioritization will be done from **customer's perspective**. Here, we will take the requirements prioritization input from 5 (fixed) users as below:

Table 6 – Matrix for prioritization of customer's perspective:

Id	Reqs	Disagree
Pr_1	$< R_3, R_2, R_1, R_4, R_5>$	6
Pr_2	$< R_3, R_2, R_1, R_5, R_4>$	6
Pr_3	$< R_1, R_3, R_2, R_4, R_5>$	6
Pr_4	$< R_2, R_3, R_1, R_4, R_5>$	7
Pr_5	$< R_2, R_3, R_4, R_5, R_1>$	9

Then, we will compute the disagreement factor by comparing the prioritization of customer & developer and counting the pairs which are different. Then, in case, any of the disagreement counts of 5 imaginary users (Pr1, Pr2, Pr3, Pr4, Pr5) are equal, we will make them unequal through the below process.

Step-1:

1. We will compare both the user prioritizations (like Pr1, Pr2) which are equal in disagreement counts.
2. We will then extract out the requirement pairs like (R1, R2) which are different.
3. In order to make the disagreement count unequal we will reverse the pairing of different pairs within the same prioritization like changing (R1, R2) to (R2, R1) and then re-calculate the disagreement count. This pairing will be reversed with the help of a user input wherein the user will be asked about the reversal through a Yes/No question in a dialog box. Only when the user approves and agrees to the reversal, we will reverse the pairs.

All in all, our objective of above step would be to make the disagreement counts of all the requirements prioritizations of 5 imaginary users as unequal.

Step – 2: In this step, we will perform the simultaneous processes of **Mutation & Crossover.**

1. Amongst the 5 user prioritizations, we take 2 prioritizations which have the lowest and second-lowest disagreement counts. For ex: if the disagreement counts of the 5 prioritizations are like **Pr1 = 5, Pr2 = 15, Pr3 = 2, Pr4 = 10, Pr5 = 6**, we take Pr3 & Pr1 and apply Mutation & Crossover operations on them simultaneously.
2. In the **Mutation operation**, we reverse the pairing of any requirement pair like changing (R1, R2) to (R2, R1).
3. In the **Crossover operation**, we fix-up a split-point in both the requirement prioritizations that we took in the above example and exchange the pairs between them. For ex: In context to the above example, if the requirements prioritization of Pr3 & Pr1 is:

Pr3: (R1, R2), (R1, R3), (R2, R3), (R2, R4), (R2, R5), (R3, R5), (R4, R5)
Pr1: (R1, R2), (R1, R4), (R2, R4), (R2, R5), (R3, R4), (R3, R5), (R4, R5)

and, we decide the split-point as 3, then after the 3rd pair, we will split the pairing and exchange the pairs as:

Pr3: (R1, R2), (R1, R3), (R2, R3, (R2, R4), (R2, R5), (R3, R5), (R4, R5)
Pr1: (R1, R2), (R1, R4), (R2, R4), (R2, R5), (R3, R4), (R3, R5), (R4, R5)

Resultant will be:

Pr3: (R1, R2), (R1, R3), (R2, R3), (R2, R5), (R3, R4), (R3, R5), (R4, R5)
Pr1: (R1, R2), (R1, R4), (R2, R4), (R2, R4), (R2, R5), (R3, R5), (R4, R5)

4. Then, we again calculate the disagreement count.

a) We perform Step – 2 till 5 iterations OR up to the point when disagreement count of any prioritization comes out to be 1 (whichever is earlier)
b) In Step – 2, if the disagreement counts of any prioritizations come out to be equal, we go to step – 1.
c) Finally after 5 iterations, we output and display the user prioritization which has the lowest disagreement count and that will be our best prioritization of the software requirements according to this Genetic algorithm.

IV. GENETIC ALGORITHM

Therefore, the resulting precedence 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 identified. Then an IGA approach to minimize the number of pairs to be elicited from the user.

The prioritization approach then **aims at minimizing the disagreement** between a total order of prioritized requirements and the various constraints that are either encoded with the requirements or that are expressed iteratively by the user during the prioritization process.

We use an interactive genetic algorithm**[IGA Approach]** to achieve such a minimization and the user whenever the fitness function cannot be computed precisely based on the information available. Specifically, each individual in the population being evolved represents an alternative prioritization of the requirements. The best individuals are then evolved into the new population by applying some mutation and crossover operators to them. For mutation, we use the requirement-pair-swapoperator, which consists of selecting two requirements and swapping their position in the mutated individual. Selection of the two requirements to swap can be done randomly and may either involve neighboring or non-neighboring requirements.

For crossover we use the cut-head/fill-in-tailand the cut tail /fill-in-head operators, which select a cut point in the chromosome of the first individual, keep either the head or the tail, and fill-in the tail (head) with the missing requirements, ordered according to the order found in the second individual to be crossed over.

The algorithm used for such prioritizing using interactive approach is as follow[IGA Approach]:

Algorithm: Compute prioritized requirements

Input *R*: set of requirements

Input $Ord_1,\ldots,Ord_k$: partial orders defining priorities and constraints upon *R* ($Ord_i \subseteq R \times R$ defines a **DAG**)

Output $<R_1,\ldots..,\ Rn>$**:** ordered list of requirements

1. Initialize *Population* with a set of ordered lists of requirements $\{Pr_i,\ldots\}$
2. *elicitedPairs* :=0
3. *maxElicitedPairs* := ***MAX*** (default = 100)
4. *thresholdDisagreements* := ***TH*** (default =5)
5. *topPopulationPerc* := ***PC*** (default5%)
6. *eliOrd* := Ø
7. **for each** Pr_i **in** *Population* **do**
8. compute sum of *disagreement* for Pr_i **w.r.t.** $Ord_1,\ldots,Ord_k$
9. **end for**
10. **while** *minDisagreemen*t > *thresholdDisagreement* ∧ *execTime*< *timeOut* **do**
11. sample *Population* with bias toward lower disagreement,e.g. using online test portal
12. sort *Population* by increasing *disagreement*
13. **if** *minDisagreement* did not decrease during last *G* generation ∧ there are ties in the*topPopulationPerc* of *Population*∧*elicitedPairs*< *maxElicitedPairs* **then**
14. *eliOrd* :=eli*Ord* ∪elicit pairwise comparisions from user ties
15. increment *elicitedPairs* by the number of elicited pairwise comparisions
16. **end if**
17. mutate *Population* using the *requirement-pair-swap* mutation operator
18. crossover *Population* using the *cut-head(tail)/fill-in-tail(head)* operator
19. **for each** Pr_i in *Population* **do**
20. compute sum of *disagreement* for Pr_i **w.r.t.** $Ord_1,\ldots,Ord_k$, *eliOrd*
21. update *minDisagreement*
22. **end for**
23. **end while**
24. return Pr_{min} , the requirement list from *Population* with minimum *disagreement*

Hence a solution is devised for prioritizing requirements based on quality based and interactive genetic approach.

V. CONCLUSIONS

The main contribution of this work is to order the requirement to improve the performance of regression tesing. In this approach two methods have been used namely quality based approach and interactive genetic algorithm. Interactive Genetic Algorithm used to produce a requirement ordering which complies with the existing priorities, satisfies the technical constraints and takes into account the relative preferences elicited from the user.The precedence of software requirements is a critical part of the analysis phase because this prioritization needs to have a base and acceptance.The system will calculate a numeric factor named as *Desirability Factor*. This factor shall determine how important any parsed requirement is. That is, lowest the desirability factor, higher the precedence of that requirement. We shall henceforth align all the requirements in the ascending order of their respective desirability factors. This set would be termed as the prioritized set of requirements according to the system.

Once the system has given its prioritization, we shall obtain the expected set of prioritized requirements from the customer and calculate a factor called *Disagreement Factor* by forming pairs. The differential pairs shall be the count of that factor. After the disagreement factor is yielded, we shall apply recursive genetic algorithm in order to minimize the disagreement count. The processes applied under this algorithm would be *Mutation & Crossover*. After the application of such processes, the final result set that we would obtain as the output shall be the best set of prioritized requirements as per system and customer.

1. This tool will help a software organization to identify the requirements clearly.
2. It will help in the furnishing of requirements in the most convenient and appropriate way possible i.e. through text-based and graphics based.
3. It will also help the organization to split and extract the requirements from a specified set of requirements easily by bifurcating it from the points of comma and full-stop occurrence.
4. It will help the organization to finally extract the best set of prioritized requirements which will enable them to understand the workflow of a software clearly and develop it efficiently.

VI. FUTURE SCOPE

1. We can further enhance his application to work on private cloud environment where several users are furnishing their requirements from several different locations
2. We may also include the principles of combining software engineering techniques like the estimation of different software metrics like Effort, Time, People, Cost etc. at the same time of software prioritization.
3. We can further facilitate this application to take real time requirements of the project combined with functional specifications and operational constraints in order to get the accurate results and best prioritization possible.
4. We can implement a hybrid form of requirements prioritization, in addition to Mutation & Crossover in order to achieve the most accurate set of software requirements prioritization.

ACKNOWLEDGMENT

We wish to take this opportunity to express heartfelt thanks to the Department of Computer Science& Engineering, KEC, Ghaziabadfor providing infrastructure and guidance to understand theparticipative precedence of software requirements using recursive genetic algorithmtechnique.

REFERENCES

[1] Carlos E. Otero, Erica Dell, Abrar Qureshi, Luis D. Otero, "A Quality-Based Requirement Prioritization Framework Using Binary Inputs" 2010 Fourth Asia International Conference on Mathematical/Analytical Modelling and Computer Simulation.

[2] Paolo Tonella, Angelo Susi,Francis Palma, "Using Interactive GA for Requirements Prioritization" 2nd International Symposium on Search Based Software Engineering.

[3] Azeem Ahmad, Aamir Shahzad, V. Kumar Padmanabhuni, Ali Mansoor, Sushma Joseph , Zaki Arshad "Requirements Prioritization with Respect to Geographically DistributedStakeholders" 2011 IEEE International Advance Computing Conference.

[4] Anil Jadhav, Rajendra Sonar "Analytic Hierarchy Process (AHP), Weighted Scoring Method (WSM), and HybridKnowledge Based System (HKBS) for Software Selection: A Comparative Study" Second International Conference on Emerging Trends in Engineering and Technology, ICETET-09.

[5] Muhammad Atif Iqbal, Athar Mohsin Zaidi, Dr. Saeed Murtaza "A New Requirement Prioritization Model for Market Driven Products Using Analytical Hierarchical Process" 2010 International Conference on Data Storage and Data Engineering.

[6] Md. Rizwan Beg, Ravi Prakash Verma, Alok Joshi"Reduction in number of comparisons for requirement prioritization using B- Tree" 2009 IEEE International Advance Computing Conference.

[7] Martin S. Feather, Steven L. Cornford, and Kenneth A. Hicks, James D. Kiper, Tim Menzies, "A Broad, Quantitative Model for Making Early Requirements Decisions" 2008 IEEE.

[8] Md. Rizwan Beg, Qamar Abbas , Ravi Prakash Verma"An Approach for Requirement Prioritization using B-Tree" 2008 IEEE.

[9] Kevin Logue and Kevin McDaid "Handling Uncertainty in Agile Requirement Prioritization and Scheduling Using Statistical Simulation" Agile 2008 Conference.

[10] Dunhui Yu, Keqing He and Jian Wang,Haiyan Mao "An Algorithm for Priority Ranking of IndividualizedFunctional Requirements in Networked Software."

[11] Steven K. Sherman "Algorithms for Timing Requirement Analysis and Generation"2008 IEEE.

[12] Nancy R. Mead "Requirements Prioritization Case Study Using AHP" 2008 Carnegie Mellon University.

[13] Andrea Herrmann*, Maya Daneva "Requirements Prioritization Based on Benefit and Cost Prediction:An Agenda for Future Research" 16th IEEE InternationalRequirements Engineering Conference.

[14] Patrik Berander , Kashif Ahmed Khan , Laura Lehtola "Towards a Research Framework on Requirements Prioritization."

