Volume-4, Issue-5, May-2015 • ISSN No 2277 - 8160

Research Paper | Technology

Iterative Precedence of Software Requirements Through Genetic Algorithm Involving System and Customer Interaction

Tejal Sharma — CSE , Mtech

Mahesh Singh — Asst. Professor CSE Department, Advanced Institute of Technology & Management

ABSTRACT

This paper represents a methodology for iterative precedence of software requirements with genetic algorithm with customer interaction . This approach maximizes the value delivered to clients and accommodates changing requirements. The proposed approach attempts to quantify the quality of requirements to provide a measurement this is representative of all quality criteria identified for a specific software project.

KEYWORDS : Requirement prioritization, iterative genetic algorithm ,desirability factor ,search techniques.

Introduction

Software industry plays an integral role in our day to day life and in growth of industry. Its presence is quotidian; people rely on software for many purposes such as safety critical system, national security , financial system etc. Every industry has a specific requirements , methods and they operate based on them. For eg, a toy company has its own methods which are prioritized according to the needs of customer. In this we aim to develop a system that takes requirements of users as input and gives the sets of prioritized requirements as output. After we specify the requirement , we parse them to determine meaningful requirement by applying quality and sub-attributes to the requirement. These are applied to the manager manually who is responsible for the delivery of project. The system in the following research paper will calculate Desirability factor (determines importance of parsed requirement) ; lower desirability higher precedence. After the system has given prioritization we calculate the Disagreement factor by forming pairs. Now we will apply mutation and crossover. The final output will give the best set of prioritized requirement .

Background work

Many techniques used in the current prioritization approach which assigns a rank to each requirement in a candidate set according to specific criteria such as value of requirement for the customer or requirement development cost. The rank of requirement can be expressed as its relative position with respect to other requirement in the set such as binary search procedure. 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 quality for that particular quality attribute. Once all requirement are evaluated the desirability factor is computed to fuse all measurement into one compact unit which is standard og all quality attribute. This compact value is computed by using a set of desirability functions that take into account consideration the priority of each quality attribute. Therefore the resulting priority of each requirement is derived from the decision makers goals for a specific software project. This result in a requirement prioritization approach based on how well requirement meet quality attributes and how those quality attributes are for the identified software project.

Though the binary scale to rank requirements is a practical approach but it is not good for features that do not lend themselves for binary assessment.

Solution approach

The prioritization approach we propose 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 to achieve such a minimization, taking advantage of interactive input from 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. When individuals having a high fitness (i.e., a low disagreement with the constraints) cannot be distinguished, since their fitness function evaluates to a plateau, user input is requested interactively, so as to make the fitness function landscape better suited for further minimization. The prioritization process terminates when a low disagreement is reached, the time out is reached or the allocated elicitation budget is over.The application has below functions:

In this functionality, the developer could type in the requirements in a text editor OR can input them through a text file. 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 .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.

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 a

s below will be generated, where 1 will signify a Yes and 0 will signify a No.

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

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

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.

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

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

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

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 6 (fixed) users as below:

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
Pr_6	$< R_2, R_3, R_5, R_4, R_1 >$	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 equ1. 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.

Process:
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.

1. 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)

2. In Step – 2, if the disagreement counts of any prioritizations come out to be equal, we go to step – 1

3. 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

al, we will make them unequal through the below process.

Conclusion
This tool will help a software organization to identify the requirements clearly.It will help in the furnishing of requirements in the most convenient and appropriate way possible i.e. through text-based and graphics basedIt 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 occurrenceIt 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.

REFERENCES

1. Requirements Prioritization Based on Benefit and Cost Prediction: An Agenda for Future Research Andrea Herrmann* , Maya Daneva+ * University of Heidelberg, Faculty of Mathematics and Computer Science, Software Engineering Group, 69120 Heidelberg, Germany1 + University of Twente, Department of Computer Science, PO Box 217, 7500 AE Enschede, The Netherlands m.daneva@utwente.nl | 2. An evaluation of methods for prioritizing software requirements Joachim Karlssona,b, *, Claes Wohlinb , Bjo¨rn Regnell c a Focal Point AB, Teknikringen 1E, SE-583 30 Linko¨ping, Sweden b Department of Computer and Information Science, Linko¨ping University, SE-581 83 Linko¨ping, Sweden c Department of Communication Systems, Lund University, SE-221 00 Lund, Sweden Received 7 February 1997; revised 5 November 1997; accepted 13 November 1997 | 3. Prioritizing Quality Requirements based on Software Architecture Evaluation Feedback Anne Koziolek Department of Informatics, University of Zurich, Switzerland koziolek@ifi.uzh.ch | 4. The Effectiveness of Requirements Prioritization Techniques for a Medium to Large Number of Requirements: A Systematic Literature Review Auckland University of Technology as a Part of the Requirements for the Degree of Master of Computer and Information Sciences November 2009 School of Computing and Mathematical Sciences | 5. http://link.springer.com/chapter/10.1007/3-540-28244-0_4 | 6. http://www.sciencedirect.com/science/article/pii/S0950584997000530 | 7. http://link.springer.com/chapter/10.1007/978-3-540-24659-6_36 |