Requirements dependencies-based test case prioritization for extra-functional properties

Muhammad Abbas[*], Irum Inayat[#α], Mehrdad Saadatmand[*], Naila Jan[#β]
[*]*RISE SICS, Research Institutes of Sweden*, Västerås, Sweden {firstname.lastname}@ri.se
[#]*SERL, National University of Computer & Emerging Sciences*, Islamabad, Pakistan
[α]irum.inayat@nu.edu.pk, [β]nailaneena@gmail.com

***Abstract*—The use of requirements' information in testing is a well-recognized practice in the software development life cycle. Literature reveals that existing tests prioritization and selection approaches neglected vital factors affecting tests priorities, like interdependencies between requirement specifications. We believe that models may play a positive role in specifying these inter-dependencies and prioritizing tests based on these inter-dependencies. However, till date, few studies can be found that make use of requirements inter-dependencies for test case prioritization. This paper uses a meta-model to aid modeling requirements, their related tests, and inter-dependencies between them. The instance of this meta-model is then processed by our modified PageRank algorithm to prioritize the requirements. The requirement priorities are then propagated to related test cases in the test model and test cases are selected based on coverage of extra-functional properties. We have demonstrated the applicability of our proposed approach on a small example case.**

Keywords—test case prioritization, requirements inter-dependencies, meta-model

I. INTRODUCTION

Software testing is a time and budget intensive process. Testing large software systems under strict deadline may require dropping some of the test cases. Ranking the test cases and selecting a subset for execution (test case prioritization and selection) plays an important role in such cases and in overall software testing process. Testing the extra-functional or non-functional properties of the software system may also require testing for some of the functional requirements since these so-called non-functional properties could be realized by implementing some of the functional requirements (for instance, implementing an encryption function to satisfy security requirements in a system). In order to be able to test for these extra-functional properties, it is essential to know what requirements to test for. Currently, very few ([1][2]) of the test prioritization approaches are tackling this problem at the requirement-level and most of the approaches do not take the requirement-level inter-dependencies into account for test case prioritization. A study showed that the use of requirement-level information for test case selection and prioritization can be beneficial [2].

A number of requirements prioritization (RP) techniques have been proposed so far and were compared with each other to find the best fit e.g., [3]. For that, the most popular evolutionary algorithm Genetic Algorithm (GA), is reportedly being used for prioritizing the requirements (e.g. [4]). Moreover, Analytic Hierarchy Process (AHP) is widely discussed in the research community for its accuracy in terms of RP [5]. Likewise, fuzzy logic-based techniques are also reportedly being used for prioritization of requirements and even for model-based trade-off analysis [6][7]. Most of the existing techniques only focus on prioritizing conflict free requirements (for instance in [6]) and in many cases, requirements interdependencies are being ignored (for instance in [8]).

Test case prioritization and selection is also growing and an open area of research. The end goal of the prioritization and selection of test cases is to effectively test the software system under budget and time constraints or to identify the test case for regression testing. A study showed that manual selection of test cases results in selection of more test cases with low bugs revealing capabilities compared to automated [9]. A number of test case prioritization and selection approaches are proposed in the literature e.g., [10][11]. Evolutionary algorithms like GA and Particle Swarm Optimization (PSO) are also being used for optimization of test case prioritization and selection process e.g., [12]. Likewise, fuzzy logic-based techniques are investigated for test case prioritization and selection [1]. Interested readers can refer to the comprehensive systematic literature review on test case prioritization and selection techniques [13].

In this paper, we address the problem of test prioritization and selection at requirement-level. The contributions of this paper are (i) a meta-model to aid modeling and visualization of requirements and their related tests, (ii) a modified page-rank algorithm for RP, and iii) an approach for prioritizing and selecting test cases for extra-functional properties based on requirement-level information. The meta-model is based on the existing concepts derived from literature for instance stakeholder, requirements and their relationships [14]. Our meta-model is capable of modeling requirements along with the interdependencies between them and other factors (like risk, cost, time to develop and business value) that are significant for RP. For RP we have used a modified Page-Rank algorithm [15]. iii) Last but not the least, we have demonstrated the applicability of our proposed approach on a small example case. The future work includes the evaluation of the proposed approach on cases from our industrial partners. The rest of the paper is structured as: Section II discusses our proposed model-based RP approach and it also discusses how the requirements priorities can be propagated to test cases as initial test priorities. Section III demonstrates the applicability of our proposed approach, and Section IV discusses our future work and concludes the paper.

II. PROPOSED APPROACH

In this study, a model-based approach is used for requirements and tests prioritization. The approach allows modeling of requirements and test related information and their relationships (e.g. depends conflicts, derives, defines, refines, realizes and tested by), stakeholder affiliation (e.g. primary, secondary), stakeholder's assigned priority (must

have, should have etc.), risk (from 1 to 10 presenting the risk level of requirements), cost (from 1 to 10), time to develop/ expected development time (from 1 to 10 value representing the time required for development), and business value (from 1 to 10 presenting the value being added to the project by the requirement). Our meta-model paves way for requirements-level test prioritization and selection. The inter-dependencies modeled in our proposed meta-model are later used by our modified Page-Rank algorithm for requirements prioritization (later used for test-case prioritization) along with other factors like risk, cost, and business value. Initial priority (`StakeHolderPriority`) is assigned to the requirements (optionally by requirements analyst) using MoSCoW technique (Must have, Should have, Could have and Would have) [16] priorities. The optional `StakeHolderPriority` also contributes to the requirements prioritization in an ordinal fashion as per the MoSCoW [16]. For instance, a `MustHave` adds 9.0, `ShouldHave` adds 6.0, `CouldHave` adds 3.0, and `WouldHave` (no value literal being assigned) adds 1.0 to the priority. The initial (optional) `StakeHolderPriority` adds the stakeholder's say to the final priorities. The `mbrpPriority` is a property that we use to capture automated priority calculation by Page-Rank algorithm. A set of test cases required to test each requirement can also modeled in our meta-model. It is important to note that at this stage only test case IDs are required from the test model and the meta-model is independent of the testing profile being used. Fig. **1** shows our meta-model developed in Ecore[1]. Note that our meta-model is still evolving and in future we will be extending it to support modeling of the non-functional requirements in more comprehensive manner. As of now it facilitates our approach for requirements-level test case prioritization and selection.

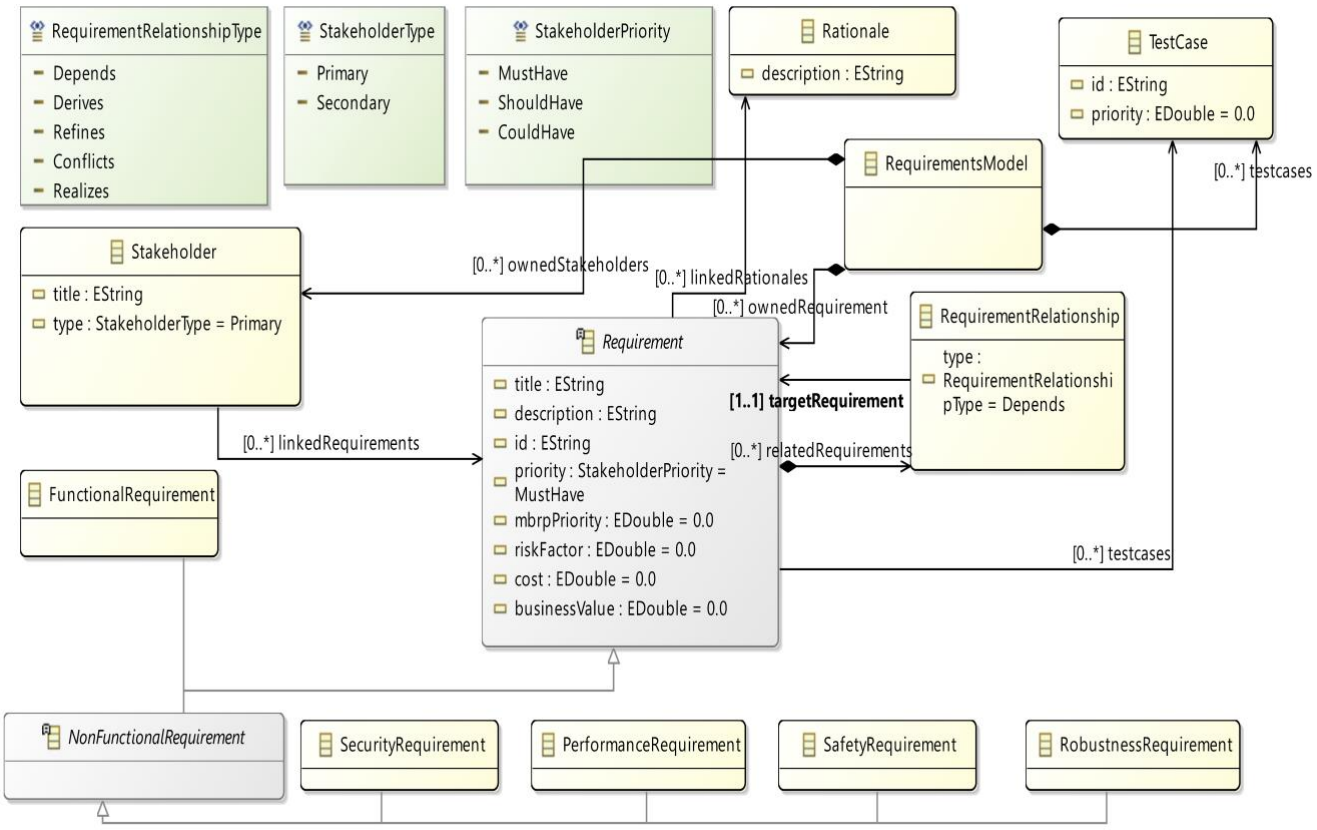

Fig. 1 Meta-model for requirements and test case prioritization

Visualization of our meta-model is aided by a Sirius[2] based concrete syntax. At the moment, we have created just one representation for our meta-model but since we have chosen a viewpoint driven representation approach, visualization can be extended for integration of risk analysis, work break-down, and even for validation plan. Our tool allows drag & drop facility for the creation of requirements, addition of stakeholders, and other relevant concepts like drawing relationships with the stakeholders and in between the requirements and test cases.

RP and test case prioritization based on requirements-level information is done by following the below-mentioned steps:

(i) Assigning an initial rank to each requirement (as per the equation (1)) that is added to overall priority of each requirement. Note that after execution of this step each requirement will have the same initial rank:

$$Rank_{initial} = (Total_{requirements} * (Total_{requirements} * 0.555)) / Total_{requirements} \quad (1)$$

(ii) Then cost of requirements is calculated using equation (2). Cost of a requirement contributes to its priority value:

$$Cost_{contribution} = Business\ Value_{req} / Cost_{req} \quad (2)$$

The contribution of risk is also essential and is calculated automatically by calculating the ratio of risk and business value. The contribution of risk in our case is calculated as per equation (3) and is also being used for calculation of Risk/Reward in different forms in the literature.:

$$RiskContribution_{req} = BusinessValue_{req} / Risk_{req} \quad (3)$$

The risk, cost, and MoSCoW contributions are calculated (as per the equations) for each requirement and are added to the overall priority of each requirement. Note that the cost, business value, and risk of each requirement (in the equations) are supposed to be modeled by the requirement analyst and are optional. If information related to risk, cost and business value is not available, it will result in purely dependencies-based prioritization (since equation 2 and 3 will just increment the priority by one).

iii) In this step, the algorithm extracts all the dependencies among inter-dependent requirements. Each inter-requirement edge (incoming) is weighted by dividing the current priority of the source requirement (of the link) equally among all links except for the conflict links. The link's contribution to the priority is calculated as per equation (4).

$$LinkContribution_{target\ req.} = (CurrentPriority_{src.\ req.} / No.\ of\ out\text{-}going\ Edges_{src.\ req}) \quad (4)$$

The link contribution is added to the overall priority if the edge is not representing a conflict. For conflict edges, half of the link contribution value is contributed to the priority of the target requirement if the priorities of the source and target requirements are equal. In case the current priority of the source requirement is higher than the target requirement the link contribution is subtracted from the priority of the target requirement.

iv) Information about test cases such as IDs from the test model and the coverage they provide to requirements (source requirement(s) of the test case represents this) are modeled. Each requirement is tested by a set of test cases and this can be model using `<tested by>` link in our instance model (see Fig. **2**). The priorities are propagated from the source requirement to the target test cases, by dividing the priority of source requirement equally into the linked test cases. Test cases providing coverage to more requirements are likely to have higher priority in our case. This step of our approach produces an ordered list of test cases based on the requirements priorities.

v) The last step of our proposed approach is the selection of test cases based on several criteria. Currently, the approach allows the selection of test cases based on extra-functional properties (for instance selection of test cases for performance, safety etc.). In our small example, we listed the results obtained from selection of test cases for security requirements. The approach also allows the selection of test cases based on coverage provided by the test cases to the requirements.

III. Demonstration of Applicability

For demonstration, we have considered a small example case shown in Fig. **2** (instance of our meta-model). We took four inter-linked functional and non-functional requirements associated with six test cases. As shown, F1 and F2 are realizing S1 (an extra-functional security requirement).

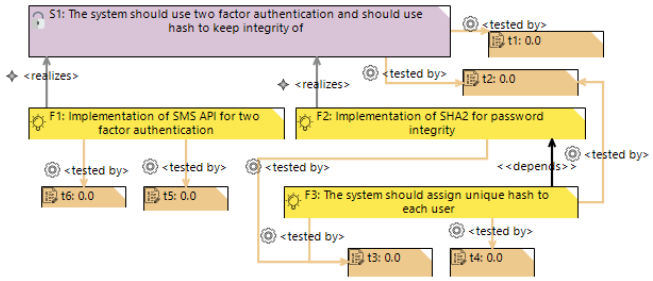

Fig. 2 Example case (Instance Model)

The S1 is tested when t1 and t2 are executed. The t2 test case also provides some coverage to the F3 functional requirement. Using the instance model in Fig. **2** (with no risk business value and cost at this point and with initial priority of `MustHave`), we have shown the results of applying first three steps of our proposed approach in TABLE **I.**

TABLE I Prioritized list from PageRank

ID	Requirements Data				
	$Rank_{sk}$	$Rank_{1}$	$Rank_{2}$	$Rank_{3}$	$Rank_{final}$
S1	9.0	11.22	12.22	13.22	39.66
F2	9.0	11.22	12.22	13.22	26.44
F1	9.0	11.22	12.22	13.22	13.22
F3	9.0	11.22	12.22	13.22	13.22

The $Rank_{sk}$ column of TABLE **I** shows the `StakeHolderPriority` (MustHave, 9 in our example case). After execution of equation (1), we ended up adding 2.22 as initial rank to all the requirements. As for this example case we did not modeled the cost, risk and business values, our approach increments the priorities by one for each equation in step ii. Thus, after execution of equation (2) and (3) our approach produces 13.22 as priority for each requirement. The step iii of our approach uses PageRank algorithm for dependencies-based ranking. Each edge in the requirements model is weighted by dividing its source requirement's priority equally in all the sibling edges. For our example case, the edge between `F1` and `S1` is weighted as: `F1`'s priority/number of out-going edges from `F1` (13.22/1 = 13.22). The weight of the edge between `F1` and `S1` results in an increment of 13.22 in `S1`'s priority. Note that as of now `S1` has a total priority of 26.44. The edge between `F2` and `S1` results in an increment of 13.22 in the priority of `S1`. For this example, our approach produced 39.66 as final priority for `S1` (as shown in $Rank_{final}$ column of TABLE **I**). The edge between `F3` and `F2` also contributes (13.22) to the final priority of `F2` as explained in this section above.

For the step iv) of our approach, the priorities from source requirements were propagated to associated test cases which produces ordered list of test cases (shown in TABLE **II**). Our approach ranks the test cases by weighting the edges between requirements and test cases and finally incrementing the test cases' priorities by the weight of the links. For our example, `t5` and `t6` got their priorities as: `F1`'s priority / no. of related tests of `F1` (13.22/2 = 6.61). The `t2`, `t3`, and `t4` gets 4.40 priority contributions from `F3`. Test cases `t2` and `t3` also get an increment of 19.83 and 26.44 to its priorities from S1 and F2 respectively. Test case `t1` only gets 19.83

[1]"EMF.", Available: https://www.eclipse.org/modeling/emf
[2]"Sirius", Available: https://www.eclipse.org/sirius/overview.html

priority from the S1. The final ranks of test cases are shown in Rank column of TABLE **II**.

For the last step of our proposed approach, the extra-functional property (security) was used as criterion for selecting test cases. Note that in our approach a tester can select any extra-functional property (available in our meta-model) to be targeted for test case selection. The ordered list of test cases is shown in TABLE **II**. Test cases IDs ending with a "*" represents that these test cases were selected by our approach for the selected criterion.

TABLE II Ordered list of Test Cases

ID	**Test Cases**		
	Type of linked requirement(s)	***Rank***	***Linked Requirements***
t3*	Functional Requirement	30.85	2
t2*	Security Requirement & Functional Requirement	24.24	2
t1	Security Requirement	19.83	1
t5	Functional Requirement	6.61	1
t6	Functional Requirement	6.61	1
t4	Functional	4.40	1

A. Results and Discussion

TABLE **I** and TABLE **II** show the results obtained after applying our approach to the small example case. The evaluation of our approach is planned two-fold. One side of the evaluation is focused on the applicability of our approach in RP and other side of the evaluation will be focused on the evaluation of the applicability of the test case selection for extra-functional properties. We verified the results on another 104 requirements' dataset from local industry of a smart home application as a pilot experiment. The list had identified dependencies (derives, depends, conflicts and refines) among different requirements. The dataset was prioritized by 30 graduate students enrolled in the course of Advanced Requirements Engineering in spring 2018 at a private sector university. The students already covered the topic of requirements prioritization in their lecturing hours Each student had an experience of working on at least one real-world project. An hour of the training session was organized before the experiment was conducted and the students were briefed about the dataset and job needed to be done. This experiment was a graded activity for the students and each submission (prioritized list) was manually evaluated by two authors of this paper. The submitted prioritized lists from students were evaluated based on the known top seven requirements (high-priority) to be in the top 20 of the prioritized lists submitted by the students. Two submissions were not complete and were not included in the final set of submissions. Total of 28 submissions were considered for this experiment. Average priorities of the 28 submissions were considered as the baseline for this experiment. To test the effectiveness of the modified PageRank algorithm in RP, we checked the normality of the prioritized lists obtained from modified PageRank. The data was not normal and we applied the Wilcoxon signed-rank test to compare the list with the base-line. The null hypothesis was rejected with the p-value 0.004. The Cohen's D effect size in our case was 0.1. Based on these results we can conclude that the modified PageRank algorithm effectively prioritized a set of requirements.

TABLE **II** shows the ordered list of test cases. After selecting the test cases for extra-functional security property, only t2 and t3 were selected. Currently, we were unable to validate the test case prioritization and selection results of our proposed approach on real-world case study (we plan to do this in future).

IV. Conclusion and Future Work

In this paper, we introduced an approach for test case prioritization based on requirements inter-dependencies. We reported i) a meta-model for modeling requirements and their related tests, ii) a modified Page-Rank algorithm for requirements prioritization and used dependencies-based priorities in test case prioritization and selection, and iii) the applicability of our proposed approach on a small example case. Our modeling tool helps to visualize requirements model, keeping in view the interdependency factor in-between the interdependent requirements and test cases. We used a modified PageRank algorithm for prioritization of extra-functional properties based on their relationships with other requirements. We then propagated the requirements priorities into test cases and selected test cases for testing extra-functional properties of the system. We evaluated our modified Page-Rank algorithm on a dataset of 104 requirements. For comparison, we also got the same list of requirements prioritized by 28 graduate students, called base-line.

Close at hand, we aim to validate the applicability of our proposed approach on a real-world case study. As another future work, we will also investigate the generality of our proposed approach for both functional and extra-functional properties (especially when the selection of test cases from a huge list is to be done for more than one extra-functional property). We are also planning to empirically evaluate our proposed approach in comparison with state-of-the-art in RP and test case prioritization and selection approaches. We are also planning to tailor our proposed approach to support test case prioritization and selection of variant intensive software systems.

Acknowledgment

This work has been supported by and received funding from the XIVT project (https://itea3.org/project/xivt.html).

References

[1] C. Hettiarachchi, H. Do, and B. Choi, "Risk-based test case prioritization using a fuzzy expert system," *Inf. Softw. Technol.*, vol. 69, pp. 1–15, 2016.

[2] J. Badwal and H. Raperia, "Test Case Prioritization using Clustering," *2013 IEEE Sixth Int. Conf. Softw. Testing, Verif. Valid.*, pp. 488–492, 2013.

[3] N. Misaghian and H. Motameni, "An approach for requirements prioritization based on tensor decomposition," *Requir. Eng.*, vol. 23, no. 2, pp. 169–188, Jun. 2018.

[4] H. Ahuja, Sujata, and U. Batra, "Performance Enhancement in Requirement Prioritization by Using Least-Squares-Based Random Genetic Algorithm," *Stud. Comput. Intell.*, vol. 713, pp. 251–263, 2018.

[5] M. Sadiq, J. Ahmed, M. Asim, A. Qureshi, and R. Suman, "More on elicitation of software requirements and prioritization using AHP," *DSDE 2010 - Int. Conf. Data Storage Data Eng.*, pp. 230–

234, 2010.

[6] F. Shao, R. Peng, H. Lai, and B. Wang, “DRank: A semi-automated requirements prioritization method based on preferences and dependencies,” *J. Syst. Softw.*, vol. 126, pp. 141–156, 2017.

[7] M. Saadatmand and S. Tahvili, “A Fuzzy Decision Support Approach for Model-Based Tradeoff Analysis of Non-functional Requirements,” *Proc. - 12th Int. Conf. Inf. Technol. New Gener. ITNG 2015*, pp. 112–121, 2015.

[8] A. Chindapornsopit and T. Samanchuen, “Requirement Prioritization for Software Release Planning Based on Customer Value with Analytic Hierarchy Process,” *ITMSOC Trans. Innov. Bus. Eng.*, vol. 01, pp. 21–27, 2016.

[9] M. Gligoric, S. Negara, O. Legunsen, and D. Marinov, “An Empirical Evaluation and Comparison of Manual and Automated Test Selection,” in *Proceedings of the 29th ACM/IEEE international conference on Automated software engineering*, 2014.

[10] A. Arrieta, S. Wang, G. Sagardui, and L. Etxeberria, “Search-Based test case prioritization for simulation-Based testing of cyber-Physical system product lines,” *J. Syst. Softw.*, vol. 149, pp. 1–34, 2019.

[11] S. Y. Shin, S. Nejati, M. Sabetzadeh, L. C. Briand, and F. Zimmer, “Test Case Prioritization for Acceptance Testing of Cyber Physical Systems: A Multi-objective Search-based Approach,” in *Proceedings of the 27th ACM SIGSOFT International Symposium on Software Testing and Analysis*, 2018, pp. 49–60.

[12] M. Tyagi and S. Malhotra, “Test Case Prioritization using Multi Objective Particle Swarm Optimizer,” in *International Conference on Signal Propagation and Computer Technology (ICSPCT 2014)*, 2014, pp. 390–395.

[13] M. Khatibsyarbini, M. A. Isa, D. N. A. Jawawi, and R. Tumeng, “Test case prioritization approaches in regression testing : A systematic literature review,” *Inf. Softw. Technol.*, vol. 93, pp. 74–93, 2018.

[14] A. Goknil and M. A. Peraldi-Frati, “A DSL for specifying timing requirements,” *2012 2nd IEEE Int. Work. Model. Requir. Eng. MoDRE 2012 - Proc.*, pp. 49–57, 2012.

[15] L. Page, S. Brin, R. Motwani, and T. Winograd, “The PageRank Citation Ranking: Bringing Order to the Web,” *World Wide Web Internet Web Inf. Syst.*, vol. 54, no. 1999–66, pp. 1–17, 1998.

[16] D. Clegg and R. Barker, *Case Method Fast-Track: A Rad Approach*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1994.