Accepted for Publication in "The 26th Asia-Pacific Software Engineering Conference (APSEC 2019)", IEEE

MBRP: Model-based Requirements Prioritization Using PageRank Algorithm

Muhammad Abbas
Research Institutes of Sweden
Västerås, Sweden
muhammad.abbas@ri.se

Irum Inayat, Naila Jan
National University of Computer & Emerging Sciences
Islamabad, Pakistan
irum.inayat@nu.edu.pk, nailaneena@gmail.com

Mehrdad Saadatmand
Research Institutes of Sweden
Västerås, Sweden
mehrdad.saadatmand@ri.se

Eduard Paul Enoiu, Daniel Sundmark
Mälardalen University
Västerås, Sweden
eduard.paul.enoiu@mdh.se, daniel.sundmark@mdh.se

Abstract— **Requirements prioritization plays an important role in driving project success during software development. Literature reveals that existing requirements prioritization approaches ignore vital factors such as interdependency between requirements. Existing requirements prioritization approaches are also generally time-consuming and involve substantial manual effort. Besides, these approaches show substantial limitations in terms of the number of requirements under consideration. There is some evidence suggesting that models could have a useful role in the analysis of requirements interdependency and their visualization, contributing towards the improvement of the overall requirements prioritization process. However, to date, just a handful of studies are focused on model-based strategies for requirements prioritization, considering only conflict-free functional requirements. This paper uses a meta-model-based approach to help the requirements analyst to model the requirements, stakeholders, and inter-dependencies between requirements. The model instance is then processed by our modified PageRank algorithm to prioritize the given requirements. An experiment was conducted, comparing our modified PageRank algorithm's efficiency and accuracy with five existing requirements prioritization methods. Besides, we also compared our results with a baseline prioritized list of 104 requirements prepared by 28 graduate students. Our results show that our modified PageRank algorithm was able to prioritize the requirements more effectively and efficiently than the other prioritization methods.**

Keywords— requirement prioritization, requirements interdependencies, meta-model, page-rank

I. INTRODUCTION

Software development is a time and budget intensive process. A successfully developed software system not only depends on its correct functioning but also depends on the value delivered to the stakeholders [1]. In today's software development culture where requirement changes are frequent and continuous, the decision of which requirements will be considered first (i.e., requirements prioritization (RP)) in a specific iteration is very important. Requirements prioritization is defined as a process of prioritizing a set of requirements based on different parameters of interest e.g., risk, cost, time and inter-dependencies [1]. The prioritized list (if performed for a software release) is developed and used with the same process being repeated for upcoming iterations. For example, the RP results can be used to plan and select the optimal set of requirements for development in the next release.

A number of RP techniques (e.g. [2][3][4]) have been proposed in the literature together with some empirical evidence on their evaluation. Genetic Algorithms (GA) is also being used for prioritizing the requirements (e.g. [5][6]) with promising results. In addition, the Analytic Hierarchy Process (AHP) is one of the most discussed prioritization approaches in the research community and is actively being used for RP [7][8][9]. The use of fuzzy logic-based techniques for prioritization of requirements has also been investigated in the literature (e.g., [9][10]). Unfortunately, most of the existing techniques are only focusing on prioritizing conflict-free requirements (for instance in [11]) and in many cases, these approaches are ignoring requirements interdependencies (e.g., [7][12][13][14]). Requirements dependencies play a huge role in the overall software engineering process and researchers have tried to compute them automatically [15].

Model-Driven Engineering (MDE) is an efficient and effective way of both managing software complexity, as well as providing a basis for the systematic development of software at various abstraction levels. MDE has been applied in Requirements Engineering (RE) for structuring, formalizing, and visualizing the requirements in the form of models (e.g. [16][17][18]). The resulting models can be used for generating design models and executable code by using Model-Driven Development (MDD) tools and technologies and are useful to aid in software analysis during the whole development process (e.g., trade-off analysis [19]). Extending the potential of using such models, one can use these for defining the dependency between requirements, with the goal of automatically performing dependency-based prioritization. A more recent work suggests that the use of the PageRank algorithm [20] for RP [11] is effective for ranking requirements based on dependencies. However, these kinds of approaches are not taking into account non-functional requirements and consider only optional requirements without any requirement conflicts. This impedes the possibility of using such RP approaches in realistic scenarios.

To alleviate the aforementioned restrictions, we proposed a meta-model based approach to facilitate the modeling, visualization, and prioritization of requirements and their related test cases [21]. The proposed meta-model borrows concepts from System Modeling Language (SysML[1]) which can be found in other meta-models in the literature (for

[1]"SysML", Available: https://sysml.org/

instance representing stakeholder information [22], requirements [17] and their relationships [17][23]). The proposed meta-model is capable of modeling requirements along with interdependencies between them and other factors (e.g., risk, cost, time to develop and business value) that are significant for RP. The meta-model is supported by a tool that aids the visualization, modeling, and prioritization of the requirements. These models are integrated with RP and we propose the use of a modified version of the PageRank algorithm in which the initial rank is assigned differently, and it can distinguish conflicting edges. To provide meaningful experimental evidence on the use of such an approach, we evaluated our proposed prioritization algorithm in terms of the following questions:

RQ1: Does the modified PageRank algorithm effectively prioritize a set of requirements?

RQ2: Does the modified PageRank algorithm efficiently prioritize a set of requirements?

Based on these research questions, our experiment compares our proposed approach with a list of 104 requirements prioritized by 28 graduate students registered in the Advanced Requirements Engineering course in a private university (onwards called the baseline).

The rest of the paper is structured as follows: in Section II we discuss the related work, while in Section III we discuss our proposed model-based RP approach. In Section IV we demonstrate our proposed model-based requirements prioritization approach on a small example case, while in Section V we evaluate our modified PageRank algorithm by comparing the results with the baseline and other algorithms. In Section VI we discuss the results of this paper, and in Section VII we discuss the threats to validity. Finally, in Section VIII we describe the limitations of our work and concluded the paper.

II. Related Work

Recently, RP techniques using AI have been proposed and deployed as a key component of an efficient and effective process. For example, the fuzzy logic and evolutionary algorithms along with traditional ones like the AHP [24] are widely used independently and in combination [25][26]. An evolutionary algorithm called Interactive Genetic Algorithm (IGA) [27] was used to prioritize forty-nine functional requirements based on a real case. The algorithm was compared with AHP to determine the reduction in pairwise comparison effort. The results showed that IGA outperformed AHP by decreasing the elicitation effort by 10%. Another GA based technique was proposed to prioritize requirements called Least-Square-Based Random Genetic Algorithm (LSRGA) and was empirically evaluated to measure its performance in comparison to IGA [5]. Gradient Descent Ranking (GDRanking) [28] is a machine learning approach for prioritization of requirements elicited through the Quality Function Deployment (QFD) [28]. This approach has two distinct phases for pairing and balancing both the functional and non-functional requirements. The proposed method was evaluated for four pairs on both functional and non-functional requirements. However, issues like requirement dependencies, renewing the requirements rank with an addition of new requirements and scalability are not considered. Another machine learning based approach called Case-Based Ranking (CB-Ranking) is proposed for requirements prioritization [29]. CB-ranking uses pair-wise comparisons (e.g., AHP) and the elicitation of the candidate priority between requirements relies on Boolean values. This gives less noisy data and concrete priority values for the pairs. Their results showed that CB-ranking is more effective and accurate as compared with AHP with an increase in the size of the dataset. Further, the experimental results showed that CB-ranking performs better than AHP by reducing the number of disagreements. Nevertheless, this work relies on a rather small data set and is not taking into account the dependency factor between requirements. Interested readers can have look at the comprehensive review on RP techniques [30].

AI techniques are helping RP by providing better support for handling non-functional RP, prioritizing a large number of requirements for large-scale software systems, and tackling the precision and accuracy issues. However, there is a need to evaluate RP techniques in more realistic situations by taking into account factors like the interdependency between requirements, and conflicting requirements. In this context, a recent study proposed an i* model-based requirement prioritization technique using the PageRank algorithm [11]. This work considered only optional and conflict-free functional requirements. To the best of our knowledge, only one recent study [11] considered the information available in the models for prioritization. Moreover, our approach seeks to improve previous approaches by taking into account other factors for requirements prioritization such as risk, business value, cost, dependencies, and even conflicts. This also makes SysML not applicable in our case. SysML lacks in taking into account the relevant factors like risk, business value, and cost. Our approach is also independent of the type (i.e., Functional/Non-Functional) of requirements being prioritized. Prioritizing requirements with conflicts can help in providing decision support for deciding the resolution of conflicting requirements based on priorities.

III. Proposed Approach

This section describes our proposed approach for requirements prioritization using PageRank algorithm and the meta-model (developed in Ecore[2]) behind the approach. We also provide a (prototype) tool[3] support that aims to support the requirement visualization, analysis, and prioritization.

Our tool supported approach allows the generation of an instance model from a spreadsheet exported from a requirements management tool (e.g., DOORS[4]). At the moment, the requirements dependencies are to be written manually by the requirement analyst in a spreadsheet or visually in our tool. We aim to automate this step by the use of natural language processing algorithms. We also considered other vital factors like risk, cost, business value and time to develop (cost) for RP. The optional factors can also be modelled manually or can be provided in the spreadsheet. The generated model is based on the meta-model shown in Fig. 1. The meta-model allows modelling of both functional and non-functional requirements with dependencies.

A. The Meta-Model and Concrete Syntax

Our requirements model borrows concepts from the SysML and other models in literature. In our case, each requirement has an `id`, `title`, `description`, `rationale` and other optional properties vital to the

[2]"EMF.", Available: https://www.eclipse.org/modeling/emf

[3]"MBRP", Available: https://github.com/a66as/mbrp

[4]"Rational DOORS", Available: https://www.ibm.com/se-en/marketplace/requirements-management

requirements prioritization process (i.e., `risk`, `cost`, and `businessValue`) can take values between one and ten representing the risk factor associated with a requirement, the expected cost for the development and value it adds to the overall project, respectively.

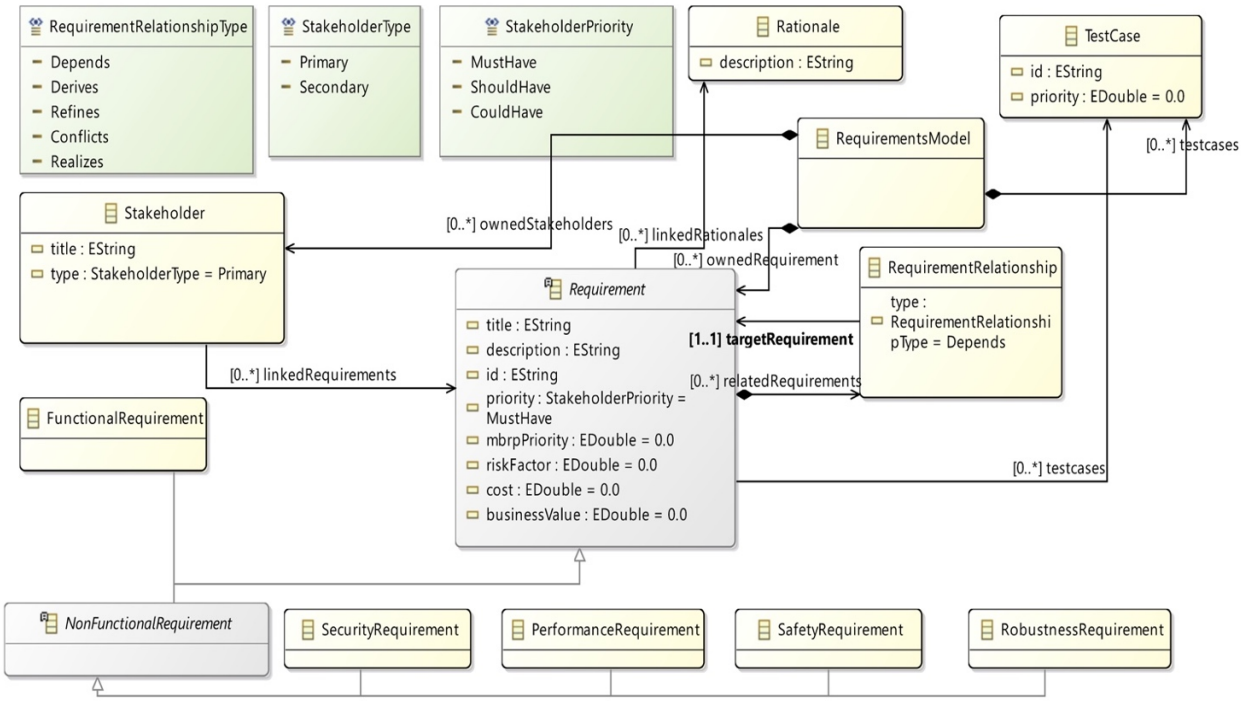

Fig. 1. Meta-Model to support requirement prioritization

An optional initial `priority` (`StakeHolderPriority`) is assigned to the requirements using MoSCoW technique (Must have, Should have, Could have and Would have)) [31] priorities. In our approach, it is recommended to model the optional properties (if available) for a better prioritization. The `mbprPriority` is used for automated priority calculation by our modified PageRank algorithm.

`Stakeholder`(s) information can also be modeled and linked to requirements. Each `Requirement` contains multiple instances of `RequirementRelationship` which can represent different types of dependencies (e.g. depends, conflicts, derives, defines, refines and realizes). A `Requirement` can also be linked to multiple test cases. Note here that it can be extended to support the requirements analysis phase in a more comprehensive way and with more factors.

We have developed a concrete syntax for our instance model in Sirius[5]. We used Sirius because it allows the generation of Eclipse-based model editors. The tools allow end-users to model the requirements and view the model visually. TABLE I. shows our concrete syntax with respect to the meta-model elements.

Functional requirements are represented using yellow notes with `id` and `title` information associated with it. Non-Functional requirements are represented using a purple note with `id` and `title` information associated with it. A `Stakeholder` is represented with a user icon. In addition, a `TestCase` is represented using orange color note with id and priority on it. The different types of dependencies are mentioned on the edges and are also color coded as follows: Depends (black arrow), Derives (grey arrow), Refines (blue arrow), Conflicts (red arrow) and Realizes (black arrow with "<realizes>" label). The association of requirements with the respective stakeholder(s) is shown through doted green lines.

TABLE I. CONCRETE SYNTAX

Meta-Model Element	Concrete Syntax		
	Representation	*Source*	*Target*
Requirement	F1:	-	-
NonFunctionalRequirement	S1: The system should	-	-
Stakeholder		-	-
TestCase	t2: 0.0	-	-
linkedRequirements		Stakeholder	Req.
Depends	<<depends>>	Req.	Req.
Derives	<<derives>>	Req.	Req.
Refines	<<refines>>	Req.	Req.
Conflicts	<<conflics>>	Req.	Req.
Realizes	<<realizes>>	Req.	Req.

B. *Requirements Prioritization*

Requirements are prioritized by following the steps shown Fig. 2. The requirement model is fed into the tool and for each requirement, the initial rank, cost contribution, risk contribution, and links contribution are calculated. All the calculated values are summed, and the sum is assigned as a priority to the requirement. These steps are repeated for each requirement and then the tool sorts the new list based on

[5]"Sirius", Available: https://www.eclipse.org/sirius/overview.html

resultant priorities producing a prioritized list of requirements. We further explain each step, in detail in this section.

The Initial Rank is assigned as shown in eq. (1). The N_{req} in the equation represents the total number of requirements in the requirements' model. The initial rank equation has the value of 0.625 representing the degree to which the priorities should be dictated by the interdependencies.

$$R\_i = N\_req \times 0.625 \quad (1)$$

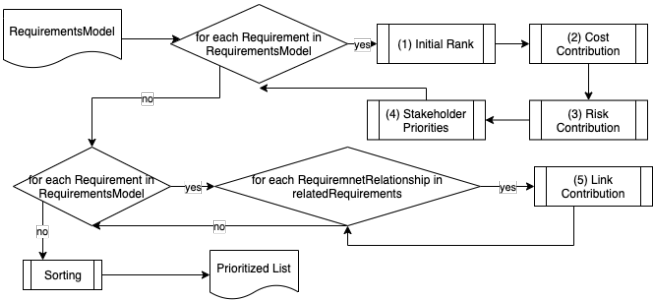

Fig. 2. Prioritization Process

The cost contribution to the priority is also calculated for each requirement and is added to the priority of the requirement. The cost contribution (to the priority) of a requirement is calculated by taking the ratio of `businessValue` and `cost` attributes of the `Requirement`. Note that if the required values for calculation of cost contribution is not modeled, the algorithm will just add one to the priority of that respective requirement.

The algorithm adds the risk contribution to the priority of the requirement. The risk contribution is calculated by taking the ratio of `businessValue` and `riskFactor` attributes of the `Requirement`. The algorithm will increment the priority value by one if the required values for the calculation of risk contribution to the priority are missing.

The optional initial stake holder's priority is also added to the priority of the requirements. These priorities are modeled using the `StakeholderPriority` enumeration in the meta-model. As in the MoSCoW method our literals also contribute to the priority in the ordinal way. The literal `MustHave` contributes 9.0 to the priority, `ShouldHave` contributes 6.0 to the priority, `CouldHave` contributes 3.0 to the priority and finally `WouldHave` contributes 1.0 to the priority. Based on the selected literal for the requirement, the targeted contribution is added to the priority of the requirement.

For the calculation of the link contributions, the algorithm extracts all the dependencies among dependent requirements. Each dependency edge (incoming) is weighted by dividing the current priority of the source (of the dependency) requirement equally among all dependency edges except for the edges representing a `Conflict`. The weighting for each dependency edge is done as shown in eq (2).

$$L_{cont.i} = Pri._{src.req.} \; / \; len(relatedReqs.) \quad (2)$$

In case of conflict edges, the contribution of the edge to the requirement priority is evaluated as per the following conditions:

- If the source requirement of the edge has a higher priority than the target of the edge, `then` half of the weight of the edge is subtracted from the priority of the target requirement.
- Otherwise, half of the weight of the edge is added to the priority of edge's source requirement.

The link contribution is added to the overall priority if the edge is not representing a conflict.

The steps of this process (mirrored as steps 1, 2, 3, and 4 in Fig. 2) are repeated for each requirement and then the links are weighted, and the links contributions are added to the priorities of each requirement. The algorithm then sorts the requirements based on the `mbrpPriority` and generates a prioritized list of requirements.

IV. Demonstration of The Proposed Approach

For the illustration of our proposed approach, we have presented a small example of a requirements dataset with four requirements. TABLE II. shows the IDs of the requirements and their relations (Link column), Business Value (BV column), cost, initial MoSCoW priority (IP column), and risk factors associated to a requirement (Risk).

Fig. 3 shows the representation of the example dataset as a model created in our tool. It shows that R1 has three incoming links, R2 and R3 have one incoming link and R4 have no incoming link. The outgoing edges are modeled as per shown the in the link column of the TABLE II.

TABLE II. Example Requirements Dataset

ID	Requirements Data				
	Link	***BV***	***Cost***	***IP***	***Risk***
R1	-	6	2	MustHave	2
R2	Depends on 1	7	5	MustHave	3
R3	Refines 2, Depends on 1	8	7	MustHave	3
R4	Depends on 1, Depends on 3	10	8	MustHave	5

All the four requirements have initial MoSCoW priority of '9' tagged as "Must Have". As per our algorithm, an initial rank is assigned to each requirement as per eq. (1). So, in this case, all the requirements have an initial rank of 2.5 (4*0.625) = 2.5). The MoSCoW priorities are then added to the initial rank. For the example dataset, all the requirements are tagged as "Must Have" and thus contributing the number 9.0 to the priority (making overall priority = 2.5 + 9.0 = 11.5, shown in Pi column of TABLE III.). After this step, risk and cost contributions are summed (shown in the CR column in TABLE III.) and are added to the overall priority of each requirement. Cost and risk contributions to the priority are calculated as discussed in Section III (e.g, for R1 cost contribution is 6/2=3 and the risk contribution is also 3 eventually resulting in a contribution of 6 to the priority).

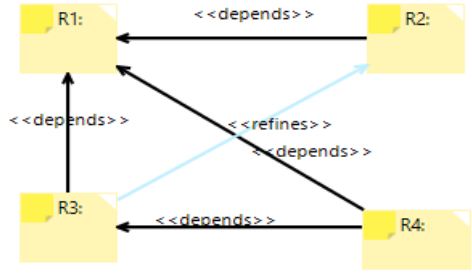

Fig. 3. Requirements model of example dataset

Till now the resulted priorities of each requirement are considered as the current priorities and are shown in Px column of TABLE III. For calculation of the links contribution to the priority, the algorithm takes requirements one by one and starts dividing the current priority in the outgoing edges. In the case of R1, there are no out-going links, so this step is skipped. In case of R2 the priority value (15.8) is assigned to the edge (R2 to R1) and is added to the priority of R1. For R3 the current priority (15.3) is divided between the two outgoing edges, incrementing R1 and R2's priority by 7.6. For R4 the current priority (14.7) is divided equally in between the two edges, resulting in an increment of 7.3 in R1 and R3's priority. The total sum of the edges contribution for each requirement is shown in EC column of TABLE III. The last column (Pf) shows the final priority of a requirement calculated by using our modified PageRank algorithm.

TABLE III. PRIORITIZED EXAMPLE DATASET

ID	Requirements Priorities				
	Pi	***CR***	***Px***	***EC***	***Pf***
R1	11.5	6	17.5	30.8	48.3
R2	11.5	4.33	15.8	7.6	23.4
R3	11.5	3.80	15.3	7.3	22.6
R4	11.5	3.25	14.7	0.0	14.7

TABLE II. shows that R4 is of the highest value to the stakeholder but since it has dependencies and thus requires R1 and R3 (that depend on R2) to be developed first. In this case, our algorithm correctly ranked the example data set of requirements.

V. EVALUATION

For evaluation of our approach, this section compares our results with existing algorithms and with a baseline (obtained from an experiment performed with human subjects). The baseline is required in our case because we wanted to evaluate our proposed approach for accuracy and efficiency. We conducted an experiment where 30 graduate students prioritized a dataset of 104 requirements. The lists from the graduate students are evaluated and an average priority (from 28 valid lists) of each requirement is considered as the baseline. the valid lists were averaged, and the baseline was obtained. The same 104 requirements dataset was then prioritized on AHP, and fuzzy Analytic Network Process (ANP), FAHP, FANP, IGA, and our modified Page-Rank algorithm. The obtained lists from the prioritization algorithms were compared with the baseline using a statistical test. Fig. 4 shows the overall flow of our experiment. The rest of the section explains each step of the experiment in more detail.

A. Preparing the Baseline

The evaluation of our proposed approach for RP is done on a dataset of 104 requirements used in the development of a smart home system. The dataset contained the required information for requirements prioritization (such as dependencies, expected development time, etc.). In order to be able to compare the results of our approach, we conducted an experiment to obtain a baseline prioritized list of requirements. The experiment was conducted in a purely academic setup where 30 graduate students participated in this activity. The students were enrolled in the "Advanced Software Requirements Engineering" course and had already worked on at least one real software development project. A one-hour session was conducted prior to the experiment where the students were briefed about the requirements' dataset. The students assigned the initial stakeholder priorities, risk, cost contributions, dependency factors as per their own understanding. For each requirement, all the mentioned values were summed, and the sum was assigned as priority to the requirement. All students prioritized the dataset, but two submissions were not completed and were not included in the baseline. Finally, a total of 28 submissions were considered for creating the baseline. The baseline list was obtained by considering the average of all the 28 priorities for each requirement.

B. Evaluation Experiment Execution

We prioritized the same dataset (used for preparing the baseline) on AHP [24], ANP [26], FAHP, FANP, IGA [27], and our modified PageRank algorithm. We selected these techniques since they are widely used for multiple criteria decision-making and deals with quantitative data. For example, AHP is a pair-wise comparison technique used for prioritization. It is definitely the most widely used and studied requirements prioritization technique.

The tool that we used for evaluating AHP is named "Super-Decision". It automates the manual input of data into models and helped us in getting the pair-wise decisions. The tool was used for AHP prioritization in many fields of business and marketing [2] and is also used for requirements prioritization especially for the multi-criteria decision-making process. We considered several optional factors: stakeholder's priority, risk, and cost. According to these factors, we prioritize the requirements dataset. The tool facilitates us to mention all these factors as criteria and all the requirements in the form of clusters and also show their relation to the mentioned factors (criteria). We created different clusters of the modules to represent the requirements as the 3rd hierarchy of the process. After creating this model, we perform the pair-wise comparisons of the requirements according to the criteria and we obtained the prioritized list of requirements.

Analytical network process (ANP) is a generalization of AHP also a multi-criteria decision-making technique used for prioritization. In ANP the number of comparisons is almost double than that of AHP because of the bidirectional relationship between the clusters. We implemented the ANP process by using the "Super Decision" tool which facilitated the bidirectional relation between the cluster and the requirements. The bidirectional arrow defines the relationship of each factor to the elements of the alternative cluster. The elements in the alternative cluster also have a dependency on the factors of criteria cluster. The tool facilitated us in all pair-wise comparisons of the clusters according to the model and allowed us to rank the comparisons. After ranking the comparison, the tool calculated the normalized priority.

We also used Fuzzy AHP which is the fuzzy version of AHP. This technique is also a multi-criteria decision-making technique and handles both qualitative and quantitative form

of data. We used the fuzzy interface system (FIS) using the Matlab ranking the dataset on Fuzzy-AHP.

Fuzzy-ANP is the advanced form of ANP just like in case of Fuzzy-AHP. For Fuzzy-ANP we created different FIS files as the clusters of Fuzzy-ANP and then implemented our dataset to get the results in the form of priorities.

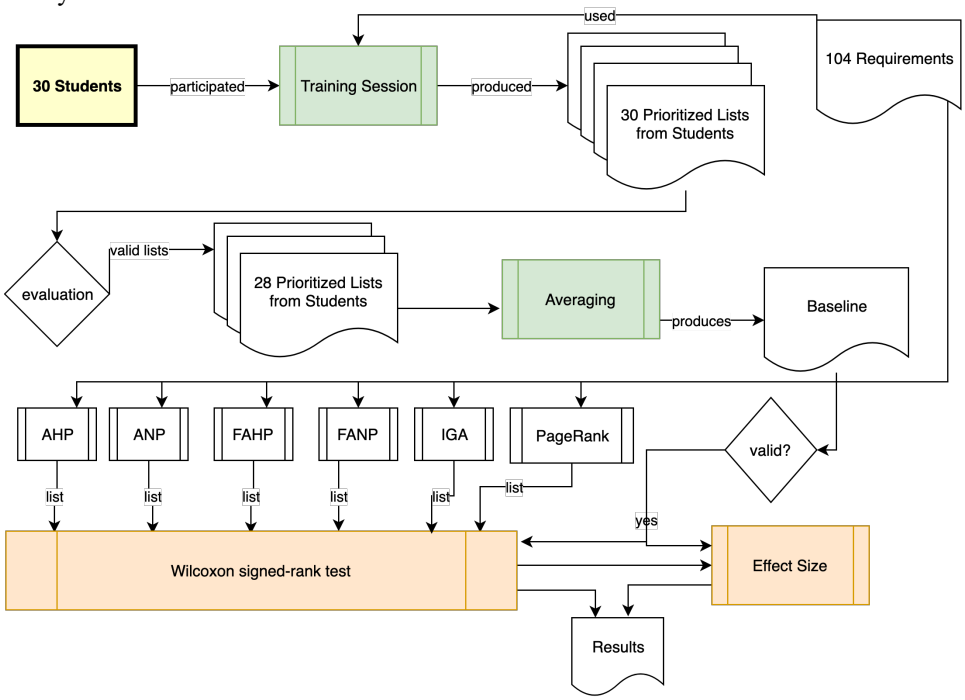

Fig. 4. Process of the evaluation experiment

The IGA was with default setup and the fitness function used for the IGA was to check if top 7 known requirements are listed in the top 20 of the obtained lists. The IGA would stop if and only if all the top 7 requirements are listed in the top of the obtained list OR it would stop if the predefined number of iterations are completed. In our case, we limited the iterations to 50,000. This took around 60 minutes on an Intel Core i3 (2.20 GHz) 2nd gen. machine with RAM of 4 gigabytes. Note that as an initial population, we provided ten lists obtained from the students.

The proposed PageRank algorithm was used to prioritize the created requirement model by using our own tool implementation. Our tool has the option of loading a requirement list from a comma-separated values file. We loaded the dataset from a file and the model was automatically generated by our tool. We manually verified the correctness of the generated model. We then prioritized the requirements model and a sorted prioritized list was generated and written to a file within seconds.

C. Experimental Results and Analysis

To answer our RQ1 (Does the modified PageRank algorithm effectively prioritize a set of requirements?) we checked the normality of the prioritized lists obtained from AHP, ANP, FAHP, FANP, IGA and modified Page-Rank. Our data observations are drawn from an unknown distribution, we applied the Wilcoxon signed-rank test [32] to evaluate if there is any significant difference between the generated RP lists and the baseline without making any assumptions on the distribution.

We found that all the lists obtained from the selected techniques (including modified PageRank) produced different results as compared to the baseline. The p-values of the applied statistical test are listed in TABLE IV. From our results, we infer that there is a statistical difference between all data sets at the 0.05 level. To check the actual difference, we applied Cohen's D effect size [33] and the result against each technique is shown in TABLE IV. Cohen D test was used to calculate two data sets and returning the deviation that a sample from one set will be different than a randomly selected sample from the other set. According to Cohen [33], the effect can be small (<0.2), medium and large (>0.8).

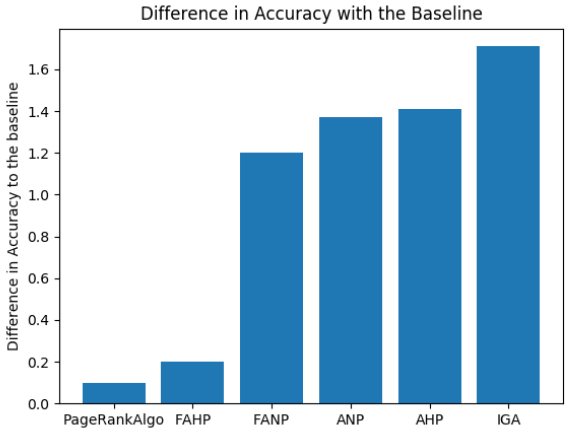

Fig. 5. Comparison of the accuracy results

TABLE IV. THE P-VALUES AND THE EFFECT SIZE

Technique	Statistical Tests and Values	
	P-Value (Wilcoxon signed-rank)	*Effect Size*
PageRank	0.004	0.1
AHP	P<2.2e-16	1.41

Technique	Statistical Tests and Values	
	P-Value (Wilcoxon signed-rank)	*Effect Size*
ANP	P<2.2e-16	1.37
FAHP	P=0.003	0.2
FANP	P=2.6e-13	1.2
IGA	P=2.9e-11	1.71

We have found that the list produced by PageRank is closer to the actual baseline and thus it can be concluded that our modified PageRank algorithm effectively prioritized the list of 104 requirements when compared to our baseline than the other techniques. Fig. 5 shows the exact differences for each technique from the baseline.

To answer our RQ2 (Does the modified PageRank algorithm efficiently prioritize a set of requirements?) we recorded the time taken by each technique to prioritize our dataset of 104 requirements. Our results show that ANP and AHP took the most time and that is because these techniques are impacted by the manual effort needed to perform them.

IGA took more than one hour to converge upon a solution, while FANP and FAHP outperformed IGA in terms of time. The FAHP technique was even more efficient in terms of time when directly compared to the FANP technique. We have also recorded the rule coding time for FANP, and the results are shown in Fig. 6.

Our proposed approach produced the prioritized list of requirements in less than a second. Note that our approach generates the requirements model automatically from the generated .csv file. Based on the time results for each technique we can clearly see that the modified Page-Rank algorithm efficiently prioritized the list of 104 requirements. The time (in minutes) taken by each technique is shown in Fig. 6.

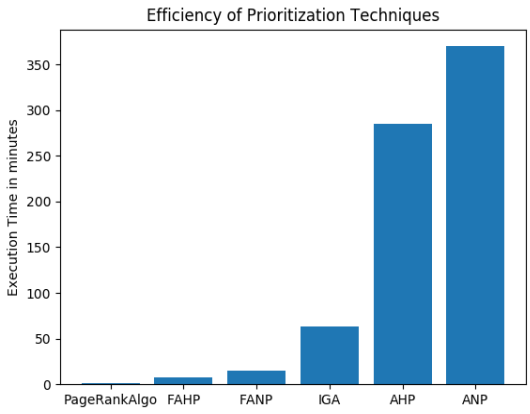

Fig. 6. Comparison of time taken by all techniques

VI. Discussion

In this paper, we used PageRank algorithm on models for RP and evaluated our approach. The results obtained from this evaluation show that all the selected techniques for the experiment produced different results when compared to a baseline. Some techniques (e.g., the Fuzzy-AHP and PageRank) was able to produce closer results to the baseline than the other techniques. Our results suggest that prioritization techniques considering requirement dependencies can produce more closer (to humans) results. In the cases where the requirements dependencies are hard to determine, automated dependencies extraction approaches can be used. Since most of the approaches have no way to represent conflicting requirements or competing stakeholder's interest, a multi-criteria decision support system with the ability to use requirement-level dependency information and conflict resolution should be considered for requirement prioritization.

In addition, the results obtained from the experimental evaluation show that manual techniques (even the tool supported) are not efficient and consume a huge amount of computational time. It is important to mention that the evolutionary algorithms might not converge upon a solution (in case of larger datasets) in a reasonable amount of time.

VII. Threats to Validity

The *internal validity* threats are related to our experimental design. The experiment was not conducted in a real industrial setup and was conducted in an academic environment. The participants of the experiments were students and not industry professionals, and this could affect the final results. Nevertheless, we ensured that the students are familiar with the requirements prioritization topic and have worked on one real software development project. While it is possible that prioritizations created by industrial engineers would yield different results, there is some scientific evidence [34] supporting the use of students in software engineering experiments.

Some *external validity* threats were addressed by selecting a diverse set of requirements. We argue that having access to a realistic dataset and rather a good number of requirements can be representative. The size of the dataset is another limitation which makes our results less generalizable for large-scale scenarios in thousands of requirements are to be consider. More studies are needed to generalize these results to other domains and RP methods used.

VIII. Conclusion

In this study, we considered dependencies for requirements prioritization with the help of a meta-model and PageRank algorithm. Our approach helps in modeling the requirements dependencies and prioritizing the requirements based on dependencies. Our approach is implemented in a tool and helps in visualizing the requirements along with dependencies. We evaluated our approach on a dataset of 104 requirements. We conducted an experiment to obtain a baseline so that we can compare the results of our algorithm and other state-of-the-art to a ground truth (baseline). We compared the results of our PageRank based prioritization with the baseline and also with five other techniques (i.e., AHP, ANP, FAHP, FANP, and IGA). We found that our modified PageRank algorithm prioritized the list of requirements effectively and efficiently and taking into account the dependency factor between requirements.

As future work, our research target includes the automated extraction of dependencies based on natural language requirements. Adding more representations and viewpoints to enhance the visualization and analysis of requirements is also one of our future concerns.

Acknowledgment

This work has been supported by and received funding from the XIVT project (https://itea3.org/project/xivt.html). This work was also supported by the Electronic Component Systems for European Leadership Joint Undertaking under grant agreement No. 737494 (MegaM@Rt2).

REFERENCES

[1] A. A and W. C., "4 Requirements Prioritization," in Engineering and Managing Software Requirements, Springer-Verlag Berlin Heidelberg, 2005.

[2] A. Görener, "Comparing AHP and ANP: An Application of Strategic Decisions Making in a Manufacturing Company," Int. J. Bus. Soc. Sci., vol. 3, no. 11, pp. 194–208, 2012.

[3] H. Wang, M. Xie, and T. N. Goh, "A comparative study of the prioritization matrix method and the analytic hierarchy process technique in quality function deployment," Total Qual. Manag., vol. 9, no. 6, pp. 421–430, 1998.

[4] S. Siddiqui, M. Beg, and S. Fatima, "Effectiveness of Requirement Prioritization Using Analytical Hierarchy Process (AHP) And Planning Game (PG): A Comparative Study," Int. J. Comput. Sci. Inf. Technol. 2013, vol. 4, no. 1, pp. 46–49, 2013.

[5] 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.

[6] P. Tonella, A. Susi, and F. Palma, "Using interactive GA for requirements prioritization," Proc. - 2nd Int. Symp. Search Based Softw. Eng. SSBSE 2010, no. February 2014, pp. 57–66, 2010.

[7] A. Chindapornsopit and T. Samanchuen, "Requirement Prioritization for Software Release Planning Based on Customer Value with Analytic Hierarchy Process," vol. 01, pp. 21–27, 2016.

[8] 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.

[9] M. A. Iqbal, A. M. Zaidi, and S. Murtaza, "A new requirement prioritization model for market driven products using analytical hierarchical process," DSDE 2010 - Int. Conf. Data Storage Data Eng., pp. 142–149, 2010.

[10] A. Ejnioui, C. E. Otero, and A. A. Qureshi, "Software requirement prioritization using fuzzy multi-attribute decision making," 2012 IEEE Conf. Open Syst. ICOS 2012, pp. 1–6, 2012.

[11] 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.

[12] N. R. Mead and S. Engineering, "Requirements Prioritization Case Study Using AHP," no. September, pp. 1–11, 2008.

[13] T. Bebensee, I. Van De Weerd, and S. Brinkkemper, "Binary priority list for prioritizing software requirements," Lect. Notes Comput. Sci., vol. 6182 LNCS, pp. 67–78, 2010.

[14] R. Beg, Q. Abbas, and R. P. Verma, "An approach for requirement prioritization using B-tree," Proc. - 1st Int. Conf. Emerg. Trends Eng. Technol. ICETET 2008, pp. 1216–1221, 2008.

[15] S. Tahvili et al., "Functional Dependency Detection for Integration Test Cases," Proc. - 2018 IEEE 18th Int. Conf. Softw. Qual. Reliab. Secur. Companion, QRS-C 2018, no. July, pp. 207–214, 2018.

[16] M. Śmiałek, W. Nowakowski, N. Jarzębowski, and A. Ambroziewicz, "From use cases and their relationships to code," 2012 2nd IEEE Int. Work. Model. Requir. Eng. MoDRE 2012 - Proc., pp. 9–18, 2012.

[17] 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.

[18] G. Mussbacher, J. Kienzle, and D. Amyot, "Transformation of aspect-oriented requirements specifications for reactive systems into aspect-oriented design specifications," 2011 Model. Requir. Eng. Work. MoDRE 2011, pp. 39–47, 2011.

[19] 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.

[20] 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.

[21] M. Abbas, I. Inayat, M. Saadatmand, and N. Jan, "Requirements Dependencies-Based Test Case Prioritization for Extra-Functional Properties," in 2019 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW), 2019, pp. 159–163.

[22] D. De Almeida Ferreira and A. R. Da Silva, "RSL-IL: An interlingua for formally documenting requirements," 2013 3rd Int. Work. Model. Requir. Eng. MoDRE 2013 - Proc., pp. 40–49, 2013.

[23] D. Blouin, E. Senn, and S. Turki, "Defining an annex language to the architecture analysis and design language for requirements engineering activities support," 2011 Model. Requir. Eng. Work. MoDRE 2011, pp. 11–20, 2011.

[24] R. W. Saaty, "The analytic hierarchy process-what it is and how it is used," Math. Model., vol. 9, no. 3–5, pp. 161–176, 1987.

[25] A. Soni, "An Evaluation of Requirements Prioritisation Methods," Int. J. Innov. Res. Adv. Eng., vol. 1, no. 10, pp. 402–411, 2014.

[26] J. Chen and Y. Yang, "A fuzzy ANP-based approach to evaluate region agricultural drought risk," Procedia Eng., vol. 23, pp. 822–827, 2011.

[27] P. Tonella, F. Palma, P. Tonella, and A. Susi, "Using Interactive GA for Requirements Prioritization Using Interactive GA for Requirements Prioritization," no. February 2014, 2010.

[28] D. Singh and A. Sharma, "Software requirement prioritization using machine learning," Proc. Int. Conf. Softw. Eng. Knowl. Eng. SEKE, vol. 2014-Janua, no. January, pp. 701–704, 2014.

[29] A. Perini, A. Susi, and P. Avesani, "A Machine Learning Approach to Software Requirements Prioritization," IEEE Trans. Softw. Eng., vol. PP, no. 99, p. 1, 2012.

[30] Q. Ma, "The effectiveness of requirements prioritization techniques for a medium to large number of requirements: a systematic literature review," Diss. Auckl. Univ. Technol., no. November, 2009.

[31] K. Brennan, A Guide to the Business Analysis Body of Knowledge, no. c. International Institute of Business Analysis, 2009.

[32] D. C. Howell, Statistical Methods for Psychology, Seventh Ed. Wadsworth, Cengage Learning, 2012.

[33] J. Cohen, Statistical power analysis for the behavioral sciences, 2nd Editio. Lawrence Erlbaum Associates, 2013.

[34] I. Salman, A. T. Misirli, and N. J. Juzgado, "Are Students Representatives of Professionals in Software Engineering Experiments?," 2015 IEEE/ACM 37th IEEE Int. Conf. Softw. Eng., vol. 1, pp. 666–676, 2015.