Hierarchical Evaluation Methodology for Software Requirements Prioritization

Vira Liubchenko, Nataliia Komleva and Svitlana Zinovatna

Odesa Polytechnic State University, 1 Shevchenko Ave., Odesa, 65044, Ukraine

Abstract

In software requirements engineering, it is often necessary to prioritize requirements. There are different criteria for evaluating requirements, reflecting the viewpoints of different stakeholders. Therefore, it is possible to apply the methods of multi-criteria decision-making, particularly the AHP method, to solve the problem of prioritization, taking into account several criteria. However, the known disadvantage of AHP is its high complexity. In this paper, we propose a methodology for reducing complexity by moving to hierarchical evaluation. We provide a case study and discuss the properties of the methodology. It is shown that the methodology can also be used for performing What-If analysis. Furthermore, the methodology is invariant to the type and form of requirements.

Keywords

Software requirement engineering, requirement prioritization, Analytical Hierarchy Process, hierarchy, pairwise comparison

1. Introduction

The product of the software development project should meet the user's expectations and end up with high-quality software. Therefore, one of the project's core phases is requirement engineering [1].

Requirement engineering is the first significant activity that involves understanding the problem domain described in a needs statement. "Requirements selection is a decision-making process that enables project managers to focus on the deliverables that add the most value to the project outcome. This task is performed to define which features or requirements will be developed in the next release. It is a complex multi-criteria decision process that has been focused on by many research works because a balance between business profits and investment is needed" [2]. Thus, one of the subprocesses in requirement engineering is requirement prioritization.

Software requirements prioritization as one of the most critical requirement engineering activities is concerned with selecting the most significant requirements from a list of voluminous requirements gathered from diverse stakeholders [3]. This process is quite complex, and many studies aim to analyze stakeholders' influence on software system's requirements engineering process [4–6]. At the same time, prioritization of requirements is equally important both when developing a new project and modifying an existing one [7]. To prioritize requirements, various techniques are used, in particular, multi-criteria decision-making techniques. One of the techniques in this group is the Analytical Hierarchy Process.

Analytical Hierarchy Process (AHP) is a decision-making method that compares unique pairs of requirements to determine which of the two compared is of higher priority [8].

In this paper, based on several publications, we identified the main issues of applying AHP to requirement prioritization. A hierarchical methodology based on AHP was proposed to solve the issues, and an information system that supports the implementation of the methodology was designed. An

Information Technology and Implementation (IT&I-2021), December 01–03, 2021, Kyiv, Ukraine
EMAIL: lvv@op.edu.ua (VL); komleva@op.edu.ua (NK); zinovatnaya.svetlana@op.edu.ua (SZ)
ORCID: 0000-0002-4611-7832 (VL); 0000-0001-9627-8530 (NK); 0000-0002-9190-6486 (SZ)
© 2022 Copyright for this paper by its authors.
Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).
CEUR Workshop Proceedings (CEUR-WS.org)

CEUR-WS.org/Vol-3132/Paper_7.pdf

example of methodology application the product backlog prioritization was considered, and the analysis of the features of the described methodology was carried out.

2. Ahp using in software requirement engineering

Although the formulation of requirements is a mandatory phase in the software development lifecycle, there is no single approach to formalizing requirements. Currently, software requirements specification, use cases, and user stories are used the most often for this purpose. In addition to the requirements specification, an estimation of their importance is usually carried out. These estimates help solve the next release problem and find the most suitable requirements to include in the next release. The core issue is to construct such estimation (or prioritization) that reflects the interests of different stakeholders. In general, these interests may not correlate. In [9], it is indicated that the "requirement prioritization process is challenging in a global software development context, where communication and coordination are core bottlenecks."

Twenty-five years ago, Karlsson first proposed applying AHP to the software engineering field [10]. Since then, many authors have published studies of the AHP implementation in the software engineering field.

The paper [11] presented the comparison of nine basic techniques for software requirements prioritization. It classified the AHP as a ratio scale prioritization technique, which produces ranked lists of requirements. In conclusion, the study author pointed out that AHP is a slow technique due to pair comparisons, which is best suited for small (less than 20) numbers of requirements. The value of AHP could be increased by combining it with other techniques or using hierarchy AHP modification.

The paper [12] justified the use of AHP for prioritizing quality attributes in software development. In particular, it said, "prioritization of quality requirements … is a useful guide for system architects". Paper, however, was limited to considering only the quality attributes without touching upon other types of requirements.

The paper [13] also pointed to the time consumption for large problems as the main disadvantage of AHP. However, at the same time, it defined AHP as "the most promising approach because it is based on a ratio scale, is fault-tolerant, and includes a consistency check."

The authors of [14] defined the excessive number of comparisons that one needs to perform as an evident limitation of AHP. As a result, they concluded, "due to its inherent complexity, AHP becomes even impractical in software projects with a large number of requirements."

As a result of empirical studies, some authors even highlighted that "people are busy to evaluate AHP on small projects and are busy in removing the limitations of AHP instead of proposing new techniques that may prove more fruitful for projects with hundreds of requirements" [15].

However, we cannot ignore the strengths of AHP. For example, the authors of [16] said, "while doing requirement prioritization thorough AHP, the participants will clearly and completely understand the requirements first, have to know the relationship amongst the requirements and criteria, under which these requirements will be prioritized." As well they noted reliable results and fault tolerance of AHP.

There were developed different solutions that kept the strengths of AHP and reduced complexity. For example, the paper [17] used the principle of pairwise comparisons of AHP to clearly present the rationality of the decision made about the prioritization of functional and non-functional requirements. The paper [18] proposed a hybrid model for requirements prioritization, which involved AHP after selecting and grouping requirements. The paper [19] considered an approach using a combination of AHP and spanning trees. However, the technology had been applied for requirements prioritization based on a single criterion (reducing delay or waiting time). To simplify the development and ranking of requirements by the authors [20] developed a UML profile for AHP. Its use allows ranking the software requirements in model-driven engineering and provides a way of ranking requirements in the model-driven context.

We concluded that AHP is appropriate for software requirement prioritization under the condition of the number of comparisons. However, there is a need for methodology, which reduces the AHP complexity when applied to medium or large projects.

3. The methodology description

The proposed methodology is based on assumptions about the form of requirements specification. We suppose that stakeholders can estimate the priorities of requirements on different aspects (or

criteria), for example, importance, time, cost, value, penalty, risk, precedence [21]. The orderings according to different criteria can be different. Therefore, it is difficult for stakeholders to determine "integral" priorities; pairwise comparisons in the AHP method should simplify the definition of such "integral" orderings.

Usually, while analyzing software requirements, business analytics or product owners define "local" priorities in accordance with one particular criterion. For example, the fragment of the typical product backlog is shown in Figure 1. Here, the product owner has defined the priorities of user stories in accordance with their importance and risk. As well, dependencies between user stories could cause the dimension for prioritization.

PROJECT BACKLOG # PRJ-PM121

e-Reader for Children AS OF 2018-02-01

ID	Name	Estimate	Importance	Risk	Dependencies
ID01	Adjustable Font size	2	Moderate	Moderate	
ID02	Connect nearby	3	Low	Moderate	
ID03	Connect with remote user	3	High	High	
ID04	Create an account	2	Moderate	Low	
ID05	Definitions	3	Moderate	Moderate	
ID06	Mobile product	1	High	Moderate	
ID07	Page maintains fidelity	3	High	High	
ID08	Pictures	2	High	High	
ID09	Rate material	2	Low	Moderate	
ID10	Read alone	2	High	Moderate	
ID11	Select material	2	High	Low	
ID12	Support voice recognition	3	Low	High	
ID13	Add to Favorites list	1	Moderate	Moderate	ID04
ID14	Add to To-Read list	1	Moderate	Moderate	ID04
ID15	Animated page turning	3	Low	Moderate	ID07
ID16	Audio Connection	3	High	High	ID03
ID17	Book covers	2	Moderate	Low	ID18
ID18	Browse all material	2	High	Moderate	ID28
ID19	Browse previously read	3	Moderate	Moderate	ID04, ID10, ID03
ID20	Connect with a friend	3	Moderate	Moderate	ID03, ID04
ID21	Delete from Favorites list	1	Moderate	Low	ID13
ID22	Delete from To-Read list	1	Moderate	Low	ID14
ID23	Friends list	2	Low	Moderate	ID20
ID24	Log into account	1	Moderate	Low	ID04

Figure 1: The fragment of product backlog

In such a case, the prioritization problem could be decomposed into a hierarchy of subproblems: prioritization of criteria and prioritizations of requirements in accordance with each criterion. Since local requirements assessments are defined, the transformation to pairwise comparisons can be automated. Additionally, it is necessary to carry out only paired comparisons of criteria. Therefore, low complexity of the procedure can be expected.

Steps involved in the prioritization procedure are:

1. Model the problem as a hierarchy containing the prioritization goal, the criteria for requirements estimation, and the requirements.

2. Establish priorities among the elements of the hierarchy first level by making a series of judgments based on pairwise comparisons of the estimation criteria.

2.a. Make $n \times n$ matrix (n represents the number of criteria).

2.b. For each pair of criteria, insert their relative intensity of importance (where the row of A_i meets the column A_j). At the same point, insert the reciprocal values to the transposed positions (e.g., cell $A_iA_j=3$, then cell $A_jA_i=1/3$).

2.c. Calculate the relative priority of each criterion.

3. Establish priorities among the elements of the hierarchy second level by calculating pairwise comparisons based on estimation under each criterion. For each criterion:

3.a. Make $m \times m$ matrix (m represents the number of requirements).

3.b. Determine the rules for converting the criterion scale into pairwise comparison estimates.

3.c. For each pair of requirements, calculate and insert their relative intensity of importance. As well, insert the reciprocal values to the transposed positions.

3.d. Calculate the relative priority of each requirement under a given criterion.

4. Synthesize these judgments to yield a set of overall priorities for the hierarchy. Estimation for each requirement is calculated as

$$E_j = \sum_{i=1}^{n} w_i e_{ij} \quad (1)$$

where w_i is a relative priority of ith criterion, e_{ij} is a relative priority of jth requirement under ith criterion.

5. Check the consistency of the result prioritization.

We have developed software for prioritizing requirements under a hierarchical methodology. Next, we describe the design of the database structure.

4. Data structure

The database is used to store both the initial data and the results of calculations.

A feature of the domain is column-oriented data, while a relational database assumes row-oriented data. Therefore, the structure of the table in Figure 1 is denormalized by the repeating group method. In addition, the number of criteria may differ for different projects, and therefore the structure of the table with the initial data may change.

Thus, a table for entering the initial data about the project is created separately for each project. To create it, the user must specify the number of criteria. In data processing, there are several transformations of the data presentation form (Figure 2).

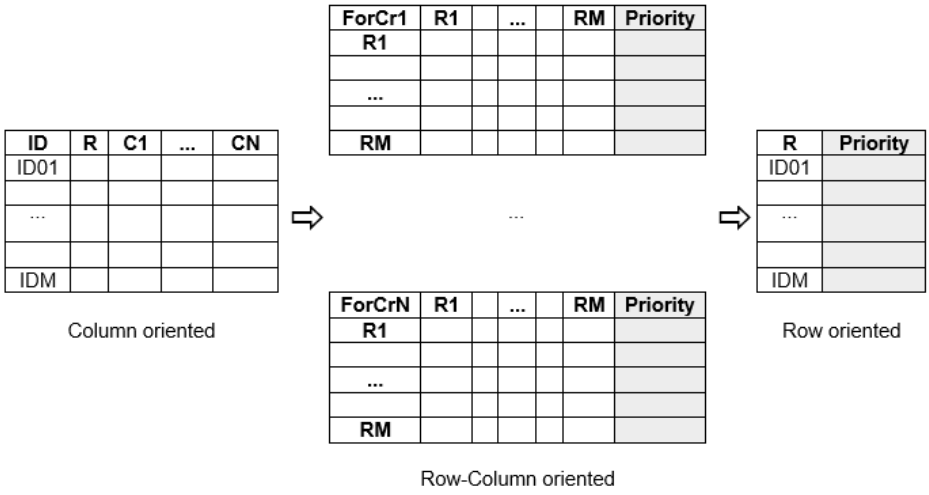

Figure 2: Table transformation

The criterion can have values of various types: categorical, numeric, multi-record. A categorical criterion involves defining a set of possible values. A numeric criterion may involve specifying the minimum and maximum values. Finally, a criterion with multiple values (dependence) involves specifying a specific subset of values from a set of possible values.

Two options are possible to obtain possible values.

1. Possible values are pre-defined by the user and then are used to fill the table of initial data.
2. The user fills in the initial data table, and the program itself forms possible values based on the entered data.

The database structure and its filling scheme are shown in Figure 3. Now we describe the general scenario of the software system actions as a sequence of steps.

1. A user creates a project and a list of criteria.
2. The system determines the number of criteria *n*.
3. The system forms a table for entering the initial data *T*0.
4. A user and the system in interactions form a list of possible values for the criteria.
5. A user fills in the table T0.
6. The system determines the number of requirements *m*.
7. The system generates a table for entering the relative priorities of the criterion *T*1; the table size is *n*×*n*.

8. A user fills in the table $T1$.
9. The system generates tables for entering the relative priorities of categorical values $T2_i$, $i \in \{1,2,...,n\}$.
10. A user fills in the tables $T2_i$.
11. The system generates tables to determine the relative priorities of requirements for each criterion $T3_i$; tables size are $m \times m$, $i = \overline{1,n}$.
12. The system calculates data for tables $T3$.
13. The system populates database tables based on content $T0$, $T1$, $T2$, $T3$.
14. The system performs calculations and enters the results into the table Requirement.

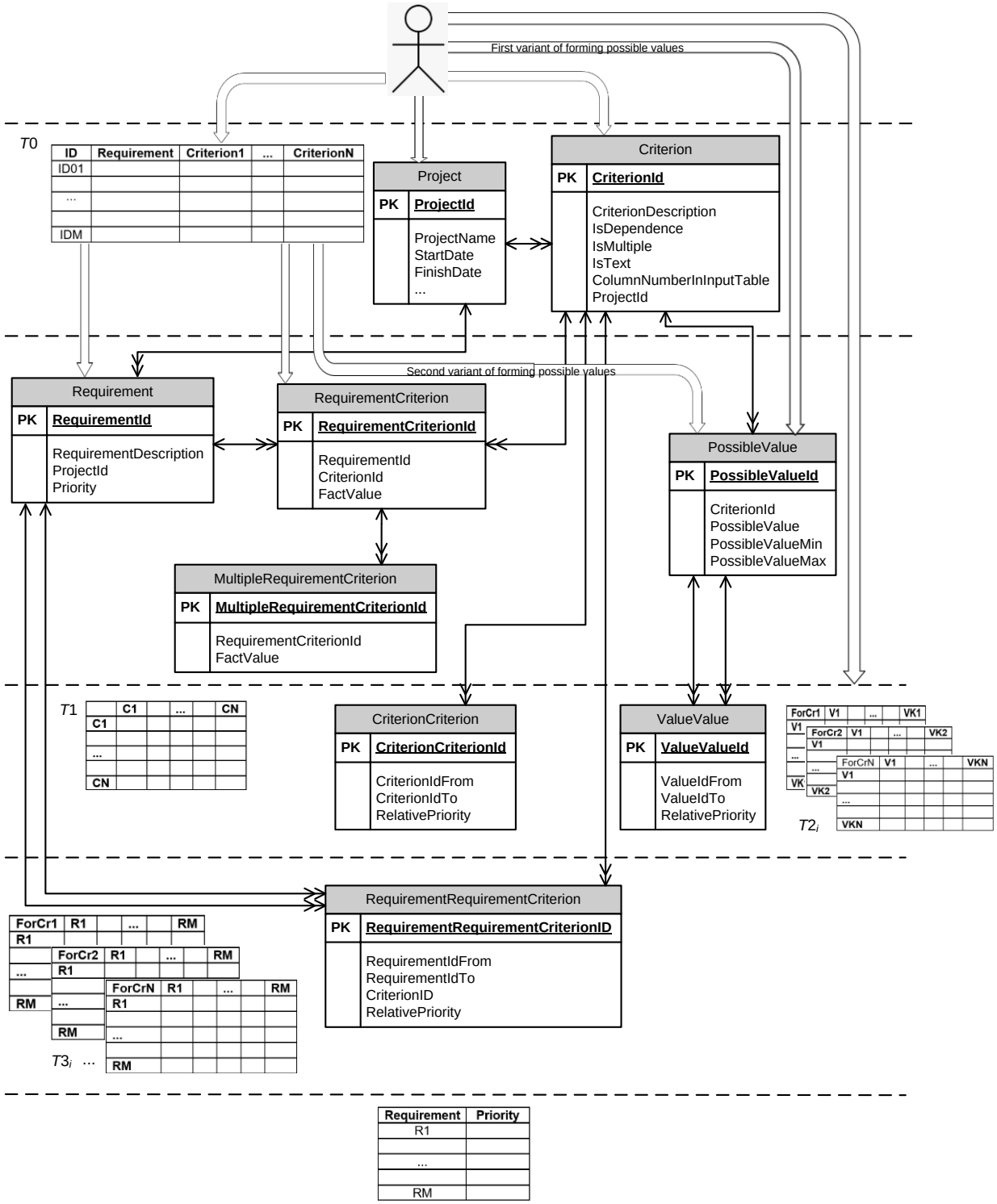

Figure 3: Database filling scheme

Thus, using a flexible data structure makes it possible to unify the methodology applied for projects with a different number or different types of criteria.

Now let us examine this methodology application by the example.

5. Case study

As an example in the paper, we have taken the product backlog presented in Figure 1. The backlog consists of 38 requirements. The hierarchical model of the problem is shown in Figure 4.

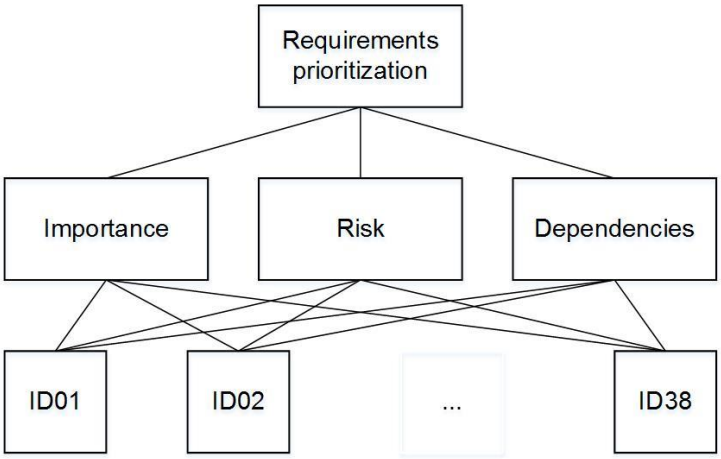

Figure 4: AHP hierarchy for product backlog

Now we should prepare the matrix of pairwise comparisons for elements of hierarchy first level. There are only three elements, and we should evaluate the relative intensity of importance in pairs "Importance – Risk," "Risk – Dependencies," and "Importance – Dependencies." The matrix definition for our example is shown in Table 1.

Table 1
Matrix of pairwise comparisons at hierarchy first level

	Importance	Risk	Dependencies
Importance	1	5	3
Risk	1/5	1	1/3
Dependencies	1/3	3	1

The contribution of each criterion to the prioritization is determined by calculations made using the priority vector. The priority vector shows the relative weights between each criterion. As a priority vector could be used the matrix eigenvector, but its exact calculation is determined only in specific cases. In our example, the priority vector is

$$\begin{aligned} \text{Importance} \quad & 0,637 \\ \text{Risk} \quad & 0,105 \\ \text{Dependencies} \quad & 0,258 \end{aligned}$$

Next, we mode to the hierarchy second level. As we see in Figure 1, the scale for criteria Importance and Risk is the same: High – Moderate – Low. Therefore, it is sufficient to determine the estimates of pairwise comparisons for different combinations of scale values (Table 2).

Table 2
Matrix of pairwise comparisons for scale values

	High	Moderate	Low
High	1	3	5
Moderate	1/3	1	3
Low	1/5	1/3	1

This transition matrix defines the transformation of requirement scores to estimations of pairwise comparison. For example, we see that the importance of ID04 is Moderated, and the importance of ID08 is High. Therefore, in the matrix of paired comparisons by the importance, we write 1/3 to the position where the row of ID04 meets the column ID08 and write 3 to the transposed position.

The criterion Dependencies needs to count in advance for each requirement the number of requirements that depend on given. These numbers become the basis for the scaling to estimations of paired comparisons.

For example, in our case, the minimum number of dependencies was 0, and the maximum number was 8. So we defined the estimation of paired comparison as $e(ID_i,ID_j)=(D_i+1)/(D_j+1)$, D_i represents the number of requirements that depend on requirement ID_i.

We do not present the results of calculating the relative priorities for each criterion that are vectors with 38 real numbers each. This is because, for the case study, these vectors do not provide additional information. Finally, we synthesized the set of overall priorities for the hierarchy in accordance with (1). We show the modified product backlog, which includes an additional column with "integral" priority for each requirement, in Figure 5.

PROJECT BACKLOG # PRJ-PM121

e-Reader for Children AS OF 2018-02-01

ID	Name	Estimate	Importance	Risk	Dependencies	Priority
ID01	Adjustable Font size	2	Moderate	Moderate		2,98
ID02	Connect nearby	3	Low	Moderate		1,84
ID03	Connect with remote user	3	High	High		10,96
ID04	Create an account	2	Moderate	Low		7,83
ID05	Definitions	3	Moderate	Moderate		2,98
ID06	Mobile product	1	High	Moderate		7,70
ID07	Page maintains fidelity	3	High	High		7,62
ID08	Pictures	2	High	High		6,28
ID09	Rate material	2	Low	Moderate		2,35
ID10	Read alone	2	High	Moderate		6,93
ID11	Select material	2	High	Low		6,15
ID12	Support voice recognition	3	Low	High		2,52
ID13	Add to Favorites list	1	Moderate	Moderate	ID04	4,70
ID14	Add to To-Read list	1	Moderate	Moderate	ID04	4,70
ID15	Animated page turning	3	Low	Moderate	ID07	1,84
ID16	Audio Connection	3	High	High	ID03	7,04
ID17	Book covers	2	Moderate	Low	ID18	2,78
ID18	Browse all material	2	High	Moderate	ID28	6,65
ID19	Browse previously read	3	Moderate	Moderate	ID04, ID10, ID03	2,96
ID20	Connect with a friend	3	Moderate	Moderate	ID03, ID04	2,96

Figure 5: The fragment of modified product backlog

The use of overall priorities simplifies the selection of requirements for implementation in the next iteration. For example, the team would have had to look for a trade-off between three criteria when working with the initial product backlog. Now the team needs to order the requirements in descending priority and get the requirement from the top of the list.

6. Discussion

Now we are going to analyze the properties of the proposed methodology.

As noted above, the main limitation of AHP use is its complexity. Building a matrix of pairwise comparisons requires performing $m\times(m-1)/2$ comparisons. This procedure should be performed by an expert and cannot be automated.

At first glance, the moving to hierarchy complicates the procedure even more. However, it is not so. At the first level of the hierarchy, one should compare only n criteria. At the second level, measurements for each criterion for assessing requirements can be performed on a categorical (nominal or ordinal) or quantitative scale. In the case of a categorical scale, it is necessary to determine pairwise comparisons for the acceptable values of the measurement scale. For quantitative scales, one should set a scaling

rule. We can assume that the upper bound for the number of criteria and the number of categorical scale values is seven plus or minus two [22]. Note that the authors have not met a case in which the number of criteria exceeded three, and the number of values exceeded five. In addition, for categorical scales within the same project, the value gradations are usually the same, making it possible to use one set of comparisons for several criteria. All these features support reducing complexity. For example, we performed 3 comparisons for the criteria and 3 comparisons for the scale values in our case. Thus, in pairwise comparisons of requirements, the number of mandatory comparisons would be larger already for $m>4$.

The next point we want to describe is the ability to perform what-if analysis. In a situation of multi-criteria decision-making, the importance of the criteria affects the final result. For example, in the requirements engineering case, criteria importance scores reflect the viewpoint of a particular stakeholder. Therefore, it can be helpful to prioritize requirements from a different stakeholder's viewpoint.

If we were to use pairwise comparisons of requirements, we would have to perform the whole procedure of comparisons and calculations for each simulated scenario. However, in the case of using a hierarchy to model different viewpoints, it is sufficient to redefine the comparisons at the first level of the hierarchy.

For example, the considered above case presented the viewpoint of the product owner. Suppose we need to model the viewpoints of the project manager and development team. The project manager considers risk as the most crucial criterion, and the development team pays attention to the dependencies between the requirements. If the relative intensity of importance can be considered approximately the same, then these viewpoints can be described by such pairwise comparisons matrices:

Table 3
Matrix of pairwise comparisons reflected different viewpoints

	Importance	Risk	Dependencies
	The viewpoint of the project manager		
Importance	1	1/3	3
Risk	3	1	5
Dependencies	1/3	1/5	1
	The viewpoint of the development team		
Importance	1	3	1/3
Risk	1/3	1	1/5
Dependencies	3	5	1

Requirements comparisons at the lowest level of the hierarchy do not need to be changed. The simulated prioritizations are shown in Figure 6.

Often software requirements are changed during the development process: new requirements could be added, existing ones could be removed or modified, requirements estimates could be revised. Obviously, in such a case, it is necessary to recalculate the priorities of requirements. Note that the recalculation would not require additional estimations since a change in the requirements does not entail a change in the importance of the criteria and comparisons of the values of the measurement scale. Therefore, the recalculation can be done automatically.

Finally, we should note that the methodology is applicable both to different types of requirements (functional, non-functional) and to different forms of recording requirements (user stories, use cases, software requirements specification).

7. Conclusion

The high complexity of the AHP method does not allow it to be used as a universal method for prioritizing requirements. The overwhelming majority of modern software products must meet dozens of requirements, and pairwise comparisons of requirements do not justify using the AHP method in the classical form.

We have proposed a methodology for reducing the complexity of the AHP method by using hierarchical prioritization scoring. The methodology is based on the assumption of the form of the technical task, which can be presented in the form of user stories, use cases, or specifications of software

requirements, with assessments of the priorities of requirements according to specific criteria. In accordance with the proposed methodology, at the first level of the hierarchy, a specialist needs to define only a matrix of paired comparisons of criteria. At the same time, at the second level of the hierarchy, requirements assessments for each criterion are calculated automatically.

ID	Name	PO scenario	PM scenario	DT scenario
ID01	Adjustable Font size	2,98	3,41	2,72
ID02	Connect nearby	1,84	2,94	2,25
ID03	Connect with remote user	10,96	10,51	12,48
ID04	Create an account	7,83	4,08	15,04
ID05	Definitions	2,98	3,41	2,72
ID06	Mobile product	7,70	5,32	7,40
ID07	Page maintains fidelity	7,62	9,15	7,77
ID08	Pictures	6,28	8,61	4,46
ID09	Rate material	2,35	3,15	3,53
ID10	Read alone	6,93	5,01	5,49
ID11	Select material	6,15	3,39	3,92
ID12	Support voice recognition	2,52	7,08	2,94
ID13	Add to Favorites list	4,70	4,10	4,58
ID14	Add to To-Read list	4,70	4,10	4,58
ID15	Animated page turning	1,84	2,94	2,25
ID16	Audio Connection	7,04	8,92	4,77
ID17	Book covers	2,78	2,20	2,52
ID18	Browse all material	6,65	4,78	5,36
ID19	Browse previously read	2,96	3,28	2,70
ID20	Connect with a friend	2,96	3,28	2,70

Figure 6: The results of the simulation

We have developed data structures that allow us to process the technical task and save the results obtained for the requirements prioritization. A feature of data structures is the ability to work with any number of criteria of categorical, numeric, or multi-record types.

The proposed methodology is especially useful for projects with the following features:

- frequent changes in project requirements (due to changes in the conditions for project development, target audience, project promotion strategy, etc.);
- impossibility of the timely formation of a complete set of requirements due to the lack of relevant expert information;
- the interest of designers in modeling options for the project development process with the influence of various stakeholders on the requirements development process.

The developed methodology and the proposed data structures can form the basis of information and analytical system for software requirements engineering management.

8. References

[1] A. Chakraborty, M. Baowaly, U. A Arefín, A. N. Bahar, The Role of Requirement Engineering in Software Development Life Cycle, Journal of Emerging Trends in Computing and Information Sciences 5(3) (2012) 723-729.

[2] J. del Sagrado, I. M. del Águila, Assisted requirements selection by clustering, Requirements Engineering 26 (2021) 167–184. doi: 10.1007/s00766-020-00341-1

[3] I. Olaronke, I. Rhoda, G. Ishaya, An Appraisal of Software Requirement Prioritization Techniques, Asian Journal of Research in Computer Science 1(1) (2018) 1–16. doi: 10.9734/ajrcos/2018/v1i124717

[4] J. Linåker, B. Regnell, D. Damian, A method for analyzing stakeholders' influence on an open source software ecosystem's requirements engineering process, Requirements Engineering 25 (2020) 115–130. doi: 10.1007/s00766-019-00310-3.

[5] A. Milne, N. Maiden, Power and politics in requirements engineering: embracing the dark side?, Requirements Engineering 17(2) (2012) 83–98. doi: 10.1007/s00766-012-0151-6

[6] D. Beimel, E. Kedmi-Shahar, Improving the identification of functional system requirements when novice analysts create use case diagrams: the benefits of applying conceptual mental models, Requirements Engineering 24 (2019) 483–502. doi: 10.1007/s00766-018-0296-z
[7] L. Gren, R. B. Svensson, M. Unterkalmsteiner, Is It Possible to Disregard Obsolete Requirements? An Initial Experiment on a Potentially New Bias in Software Effort Estimation, in: 2017 IEEE/ACM 10th International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE), IEEE Press, 2017, pp. 56–61. doi: 10.1109/CHASE.2017.10
[8] T.L. Saaty, The Analytic Hierarchy Process, McGraw-Hill, Inc., New York, NY, 1980.
[9] N. M. Minhas, A. Majeed, J. Borstler, T. Gorschek, SWVP - A Requirements Prioritization Technique for Global Software Development, in: 45th Euromicro Conference on Software Engineering And Advanced Applications (SEAA), IEEE Press, 2019, pp. 1–9. doi: 10.1109/SEAA.2019.00010
[10] J. Karlsson, Software requirements prioritizing, in: Proceedings of the Second International Conference on Requirements Engineering, IEEE Press, 1996, pp. 110–116. doi: 10.1109/ICRE.1996.491435
[11] M. Vestola, A comparison of nine basic techniques for requirements prioritization, 2010. URL: http://www.mvnet.fi/publications/software_development_seminar.pdf
[12] M. Kassab, N. Kilicay-Ergin, Applying analytical hierarchy process to system quality requirements prioritization, Innovations in Systems and Software Engineering 11 (2015) 303–312. doi: 10.1007/s11334-015-0260-8
[13] S. Siddiqui, M. Rizwan Beg, S. Fatima, Effectiveness of requirement prioritization using Analytical Hierarchy Process (AHP) and Planning Game (PG): a comparative study, International Journal of Computer Science and Information Technologies 4 (1) (2013) 46–49.
[14] J. M. Fernandes, S. P. Rodrigues, L. A. Costa, Comparing AHP and ELECTRE I for prioritizing software requirements, in: 2015 IEEE/ACIS 16th International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD), IEEE Press, 2015, pp. 1–8. doi: 10.1109/SNPD.2015.7176282.
[15] M. I. Babar, M. Ramzan, S. A. K. Ghayyur, Challenges and future trends in software requirements prioritization, in: International Conference on Computer Networks and Information Technology, IEEE Press, 2011, pp. 319–324. doi: 10.1109/ICCNIT.2011.6020888.
[16] J. A. Khan, I. Ur Rehman, Y. H. Khan, I. J. Khan, S. Rashid, Comparison of Requirement Prioritization Techniques to Find Best Prioritization Technique, International Journal of Modern Education and Computer Science 7(11) (2015) 53–59, 2015. doi: 10.5815/ijmecs.2015.11.06
[17] F. Fellir, K. Nafil, R. Touahni, System requirements prioritization based on AHP, in: 2014 Third IEEE International Colloquium in Information Science and Technology (CIST), IEEE Press, 2014, pp. 163–167. doi: 10.1109/CIST.2014.7016612.
[18] A. R. Asghar, S. N. Bhatti, A. Tabassum, S A. Ali Shah, The Impact of Analytical Assessment of Requirements Prioritization Models: An Empirical Study, International Journal of Advanced Computer Science and Applications 8(2) (2017) 303–313. doi: 10.14569/IJACSA.2017.080240
[19] M. Yaseen, A. Mustapha, N. Ibrahim, Prioritization of Software Functional Requirements from Developers Perspective, International Journal of Advanced Computer Science and Applications 11(9) (2020) 210–224. doi: 10.14569/IJACSA.2020.0110925
[20] T. Zahoor, F. Azam, M.W. Anwar, B. Maqbool, H.A. Javaid, A UML Profile for Software Requirements Prioritization, in: IEEE 10th Annual Information Technology, Electronics And Mobile Communication Conference (IEMCON), IEEE Press, 2019, pp. 885–891. doi: 10.1109/IEMCON.2019.8936265
[21] S. Hatton, Early Prioritization of Goals, in: J.-L. Hainaut, E. A. Rundensteiner, M. Kirchberg, M. Bertolotto (Eds.), Advances in Conceptual Modeling – Foundations and Applications, volume 4802 of Lecture Notes in Computer Science, Springer, Berlin, Heidelberg, 2007, pp. 235–244. doi: 10.1007/978-3-540-76292-8_29
[22] G. A. Miller, The magical number seven, plus or minus two: Some limits on our capacity for processing information, Psychological Review 63(2) (1956) 81–97. doi: 10.1037/h0043158