Journal of Systems and Software 158 (2019): 110417

DOI: https://doi.org/ 10.1016/j.jss.2019.110417

Toward Collisions Produced in Requirements Rankings: A Qualitative Approach and Experimental Study

Luis A. Rojas[a] and José A. Macías[b]

[a]Luis Alberto Rojas Pino. Ph.D. in Computer Science. Universidad Andres Bello. Facultad de Ingeniería. Antonio Varas 880. 7500971 Santiago. Chile. Phone: +56 2 27708861. E-mail: luis.rojas@unab.cl.

[b]José Antonio Macías Iglesias. Ph.D. in Computer Science. Universidad Autónoma de Madrid. Escuela Politécnica superior. Tomás y Valiente 11. 28049 Madrid. Spain. Phone: +34 914976241. E-mail: j.macias@uam.es (**corresponding author**).

Abstract

Requirements prioritization is an important issue that determines the way requirements are selected and processed in software projects. There already exist specific methods to classify and prioritize requirements, most of them based on quantitative measures. However, most of existing approaches do not consider collisions, which are an important concern in large-scale requirements sets and, more specifically, in agile development processes where requirements have to be uniquely selected for each software increment. In this paper, we propose QMPSR (Qualitative Method for Prioritizing Software Requirements), an approach that features the prioritization of requirements by considering qualitative elements that are related to the project's priorities. Our approach highlights a prioritization method that has proven to reduce collisions in software requirements rankings. Furthermore, QMPSR improves accuracy in classification when facing large-scale requirements sets, featuring no scalability problems as the number of requirements increases. We formally introduce QMPSR and then define prioritization effort and collision metrics to carry out comprehensive experiments involving different sets of requirements, comparing our approach with well-known existing prioritization methods. The experiments have provided satisfactory results, overcoming existing approaches and ensuring scalability.

Keywords: Requirement Prioritization, Requirement Collision, Qualitative Prioritization Method.

1. Introduction

Requirements prioritization is a core activity intended to elicit essential requirements for software development (Pohl & Rupp, 2011). In general, this task is manually achieved as part of

activities related to requirements engineering. Currently, even though there are several prioritization methods to systematize this task (Achimugu, et al., 2014), there are still challenges and lack of evidence about their validation (Daneva, et al., 2014).

Different authors (Curcio, et al., 2018; Daneva, et al., 2014; Achimugu, et al., 2014; Babar, et al., 2011) identified specific challenges that prioritization methods should face in order to carry out an efficient and systematic prioritization of requirements. These challenges are mainly related to key factors such as the way a method behaves when the number of requirements increases (i.e., scalability). It was also identified that, most of the time, results obtained from prioritization methods do not exactly match the expected requirements ranking desired by stakeholders. This is due to the fact that most of prioritization methods operate considering quantitative information only, which may result in a conflict of interests with respect to the stakeholders' perspective (Babar, et al., 2011). At the same time, this also generates obstacles related to the lack of accuracy on estimations (Curcio, et al., 2018).

Another important concern when classifying and prioritizing requirements automatically is the probability of producing collisions. Collisions have been barely addressed in the prioritization of requirements (Gaur & Soni, 2010). In general, overlapping problems, mainly related to quality assurance fails, are much more common and easier to find in software engineering literature (Alshazly, et al., 2014; Zalazar, et al., 2017; Zhang, et al., 2014). However, collisions comprise a quite different and technical problem that arises from the execution of the algorithms used to classify and prioritize software requirements. Nevertheless, in other disciplines, collisions are meant as a well-known problem that occurs naturally in the creation of rankings and prioritized sets of any domain using automatic algorithms (Cantu, et al., 2017; Kwon, et al., 2016; Zhou & Liao, 2012). In the case of requirements prioritization, a collision occurs when two or more requirements have the same prioritization value in a ranking, which implies that the automatic prioritization method should be in charge of minimizing or resolving this problem. Requirements collisions should not be confused with requirements conflicts or conflicting requirements, mainly addressed by the requirements engineering (Pohl, 2010).

Requirements collisions should be addressed conveniently in order to maintain the effectiveness of the automatic prioritization approach. In general, most of the existing prioritization methods do not address this issue, and they experiment important collision problems (as will be evaluated later on in Section 6) due to their quantitative nature. In this way, it becomes important to automatically discern the relevance level (priority) in requirements sets, enhancing prioritization for decision-making and verifying the consistency and accuracy of assessments in rankings as the number of requirements increases.

1.1 Contributions

The aim of this paper is to address the aforementioned drawbacks by introducing a novel prioritization method. We have considered previously published research on requirements prioritization (see Table 1), in order to build upon the knowledge already consolidated in the field and spotlight our contribution.

More specifically, the main contributions presented in our work are the following:

- The development of QMPSR (Qualitative Method for Prioritizing Software Requirements). Our method drives the prioritization process through the software project's most relevant aspects and elements. This allows to discern the relevance level of the requirements involved, obtaining a prioritized classification to reduce collisions.
- The development of a set of experiments for the evaluation of QMPSR. More specifically, we have defined two metrics: prioritization effort and collision, and we carried out 9 experiments on those metrics to assess our approach against 6 well-known existing prioritization methods: MoSCoW (Moran, 2015), Value-Oriented (Azar, et al., 2007), Wiegers (Wiegers & Beatty, 2013; Wiegers, 1999), Product Definition (Fraser, 2002), AHP (Saaty, 2008; Saaty, 1980) and Kano (Kim, et al., 2017; Kano, et al., 1984). To carry out this task, an experimental framework has been defined. The results obtained have proven that QMPSR outperforms uniformly all the prioritization methods featured, reducing collisions and improving the ranking accuracy in large-scale sets of requirements without presenting scalability problems. Experiments demonstrated that the proposed method generates less collided requirements as the number of prioritization dimensions and requirements increase. Likewise, results provided evidence for the validation of QMPSR.

Our approach addresses in a novel way the following issues related to requirements prioritization that can be considered relevant in the area:

- It implicitly specifies the relevance of a set of requirements in order to reduce the number of iterations in the negotiation phases among practitioners – i.e., requirement engineers and stakeholders (clients and users in most cases). This is especially useful in agile processes where quick decisions have to be made. In general, this aspect is sometimes overlooked, and it is commonly solved in other approaches by reiterating the evaluation or proposing an informal negotiation among decision-makers.
- While other prioritization methods are mainly focused on efficiency or capacity of finding several solutions in Pareto's optimal fronts, our method addresses the identification of different levels of evaluation for the requirements that help decision-makers focus on relevant information during the prioritization process. This allows a deeper analysis of the different aspects to obtain the priority for a given requirement.
- It researches on requirements collisions, which have been barely tackled in the literature about software requirements, from an experimental perspective.

Table 1

Analysis of previously published research on requirements prioritization methods.

Reference	Summary of the issues addressed	Issues contributed by our approach
Achimugu et al (2014)	This work identifies the difficulties that most proposals have when prioritizing requirements, such as scalability and exclusive usage of quantitative	These issues have been tackled by our approach, increasing scalability by using qualitative elements in order to clarify inconsistencies and requirements

	information.	collisions during the prioritization process.
Pergher and Rossi (2013)	The scalability issue has been also pointed out by these authors, together with the necessity of researching other variables related to the number of errors that can affect the final result of the prioritization.	Our approach investigates requirements collisions, which have not been analyzed previously.
Riegel and Doerr (2015)	Authors identified what aspects (prioritization criteria) should be taken into consideration in order to determine the value of the requirements by means of the identification and definition of the project's relevant aspects.	In order to discern the relevance level of requirements, we found out that software projects should consider mainly, but not exclusively, aspects such as *Usability*, *Strategy-Related Benefit*, and *Business Value*. In real settings, these aspects are indicated in a generic way and collectively negotiated among decision-makers according to each project's main guidelines. More specifically, our approach allows to deal with an unlimited number of aspects that can be defined for a concrete project under demand.
Pitangueira et al. (2013)	This research proposes to increase the scale of the experimentation by using larger sets of requirements. Authors also highlight the importance of stakeholders participation. This issue comprises an important factor that can lead to have better results closer to the stakeholders' requirements.	Our experimental framework analyzes the prioritization methods' behavior in large sets of requirements. Results demonstrate that our method behaves better with larger requirements sets. On the other hand, our proposal provides mechanisms to include the stakeholder's opinion through the formal definition of aspects and elements associated with the stakeholders' particular interests.
Ma (2009)	The author states that most medium-sized studies utilize a subjective improvement measure based on the user's perception.	We analyzed this issue in our work, providing mechanisms to reduce subjectivity and improve objectivity.
Herrmann and Daneva (2008)	This work studies the specification and estimation of cost and benefit in software prioritization. This work also reports on the importance of researching on the requirements' dependency.	The specification and estimation of cost and benefit have certain relationship with our activities, as we consider aspects and elements that will be used to prioritize the requirements. Regarding the requirements' dependency, the association factor in our approach can be seen as a way of addressing dependencies among requirements, identifying the elements associated with the requirements and providing an association factor to weight the elements that have more related requirements.

1.2 Methodology and Structure

The research method followed to conduct our work has been inspired by Design Science applied to information systems (Hevner, et al., 2004), which provides a framework to present and evaluate design-science research in the field. In this way, we followed the seven principal guidelines stated by Design Science. First, we have created an innovative purposeful artifact (guideline 1) based on a new algorithm for requirements prioritization in a specific problem domain, which is the reduction of collisions (guideline 2). The solution proposed is suitable for

the problem stated (guideline 3), as it has been demonstrated through the evaluation carried out, which provided evidence of being more effective than existing approaches (guideline 4). On the other hand, the proposal has been formally described using mathematical descriptions and internal properties (guideline 5), providing a search process whereby the problem space is constructed and the prioritization mechanism is posed to find an effective solution through the comparison with different prioritization methods (guideline 6). Finally, both the solution and results have been communicated through this paper to a technical and managerial audience (guideline 7).

Accordingly, this paper is organized as follows. Section 1 introduced the motivation of our work through the problem stated and the solution proposed. Section 2 presents related work. Section 3 describes, in a formal way, QMPSR, our proposed prioritization method. Section 4 presents an example of application through a specific scenario. Section 5 introduces a framework intended to carry out experimental evaluations with different prioritization methods. Section 6 describes the experiments accomplished to evaluate QMPSR against other existing prioritization methods, also reporting on the results obtained. Section 7 reports on threats to validity (Wohlin, et al., 2012; Wieringa, 2014), while section 8 describes the limitations of our method. Finally, section 9 presents conclusions and future work.

2. Related Work

Currently, there is a great variety of prioritization methods used in traditional and agile software development processes (Schön, et al., 2017; Pitangueira, et al., 2015; Achimugu, et al., 2014). Among these methods, the Analytic Hierarchy Process (AHP) method (Saaty, 2008; Saaty, 1980) is one of the most popular and cited (Achimugu, et al., 2014). AHP is used in the field of requirements prioritization to identify the priority of each requirement through a pairwise comparison matrix. The use of AHP is widespread due to its ease of application and structure, as well as its intuitive way of computing (Ishizaka & Labib, 2009). This is one of the reasons why there are several prioritization methods based on AHP.

For example, the Power Analytic Hierarchy Process (PAHPT) method (Ibrahim & Nosseir, 2016) is based on AHP, and it deals with specific issues such as power, which is of interest for some stakeholders. The Cost–Value method (Karlsson & Ryan, 1997) also uses AHP to prioritize requirements based on their perceived value and the implementation cost. Similarly, the Pairwise Analysis method (Karlsson, 1996) is based on AHP, and it prioritizes requirements by comparing each pairs to determine the requirement to be selected. The Case-Based Ranking (CBRank) method (Perini, et al., 2013) is also influenced by AHP, but it uses a machine learning technique that reduces human effort in the input information required, keeping up the accuracy of the final ranking.

In a similar way, DRank (Shao, et al., 2017), Fuzzy AHP (Lima, et al., 2011) and Hierarchy AHP (Karlsson, et al., 1998) methods utilize AHP and machine learning techniques in order to reduce the required number of pairwise comparisons. DRank presents improvements with respect to CBRank, considering the stakeholders' preferences and dependencies between requirements.

Fuzzy AHP carries out prioritization through fuzzy goals and weights for ranking requirements, dealing with ambiguous and vague data. In the Hierarchy AHP, stakeholders propose a fixed set of requirements, ranging from general to specific, in order to be prioritized.

Similarly, the Interactive Genetic Algorithm (IGA) method (Tonella, et al., 2013) is also based on AHP, featuring however a genetic algorithm to reduce the number of elicited pairs and thus obtaining the user's knowledge about the relevance level for each pair of requirements. Finally, the Cognitive-Driven method (Carod & Cechich, 2010) combines the utilization of AHP and cognitive psychology to evaluate the ability of stakeholders regarding the suggested software before the requirements prioritization process starts.

All in all, one of the most common limitations of the methods mentioned above (AHP, PAHPT, Cost–Value, Pairwise Analysis, CBRank, DRank, Fuzzy AHP, Hierarchy AHP, IGA and Cognitive-Driven) is that they do not specifically address requirements collisions throughout the prioritization process. In fact, they turn out to be time-consuming for large sets of requirements, being hardly scalable (Karlsson, et al., 1998; Hudaib, et al., 2018) in the long run as well. Besides, most of existing methods rely on prioritization mechanisms based on individual evaluation of requirements through pairwise comparisons. This misses a global understanding of the project's relevant aspects in order to guide the requirements prioritization process, also missing qualitative elements for helping reduce requirements collisions.

Carrying on with the analysis of the prioritization approaches, the Quality Functional Deployment (QFD) method (Franceschini, 2016; Crow, 1994) is the second most-cited prioritization approach (Achimugu, et al., 2014). It uses a matrix to represent the stakeholder's needs and expectations. Another similar approach is the Correlation-Based Priority Assessment framework (CBPA) method (Liu, et al., 2006), which utilizes a relationship matrix to prioritize requirements coming from multiple stakeholders, thus considering inter-perspective relationships in requirements. The Kano method (Kim, et al., 2017; Kano, et al., 1984) also allows prioritizing requirements based on the stakeholders' preferences. Nevertheless, it is focused on the characteristics of the product's differences.

Similarly, the Lanchester Theory method (Fehlmann, 2008) uses a quantitative model to drive the requirements prioritization, where requirements are prioritized according to business objectives and market share. Likewise, the Wiegers method (Wiegers & Beatty, 2013; Wiegers, 1999) estimates the relative priorities of requirements through a scheme based on the QFD concept of customer value. Finally, the Product Definition method (Fraser, 2002) prioritizes requirements taking into consideration the users' perspective, technology and business, as well as encouraging the involvement of stakeholders, user experience experts and technical analysts.

In general, the aforementioned prioritization methods (QFD, CBPA, Kano, Lanchester Theory, Wiegers and Product Definition) allow capturing diverse elements to obtain a global conception of the project (business objectives, client expectations and stakeholder needs) to drive the prioritization of requirements. However, such methods are mostly suitable for small sets of requirements due to scalability problems (Avesani, et al., 2005). Similarly, they also have

limitations regarding the subjective use of ordinal scales and ratings (Achimugu, et al., 2014). This makes it difficult to provide objective qualitative elements in order to clarify inconsistencies and requirements collisions during the prioritization process.

On the other hand, the Binary Search Tree (BST) method (Anand & Dinakaran, 2017; Hopcroft, 1983) features a binary tree where the nodes are labeled with requirements in a hierarchical order (parent-child relationship). Similarly, the B-Tree method (Beg, et al., 2009) also allows to organize the requirements in nodes that are compared to establish the relevance level using a weighted scale, thus reducing the number of comparisons.

Nevertheless, these prioritization methods (BST and B-Tree) present some limitations related to scalability problems (Aasem, et al., 2010) and the absence of priority values for the final ranking of requirements (Duan, et al., 2009). Additionally, the assessments achieved in requirements prioritization are neither based on qualitative measures nor related to relevant aspects that define the project's priorities. This makes it difficult to analyze requirements collisions and the precision of the final ranking.

It is also possible to identify prioritization methods specifically used in agile software development, such as Planning Game, MoSCoW, $100 Allocation, Dot Voting, Value-Oriented, Multi-Voting System, Round-the-Group Prioritization, Theme Screening and Weighted Criteria Analysis (Curcio, et al., 2018; Racheva, et al., 2008; Racheva, et al., 2010). To focus on the most important ones, the Moscow method (Moran, 2015) prioritizes those requirements with a higher value for the system, while the Value-Oriented method (Azar, et al., 2007) evaluates requirements according to core business and the stakeholders' values. These prioritization methods are oriented to small sets of requirements, facilitating their application by dynamic development teams. Nonetheless, in most cases, these prioritization methods are mainly focused on requirements classification, overlooking dependencies among requirements and missing a collisions management. Moreover, they also suffer from scalability problems as the number of requirements increases (Aasem, et al., 2010; Hatton, 2008).

In a nutshell, it is possible to affirm that existing prioritization methods do not consider nor evaluate requirements collisions in order to validate and improve the ranking accuracy. In addition, the project's priorities are ignored or not formally captured to verify the consistency of the principal concerns related to requirements. Finally, most of existing approaches are based on quantitative measures, undergoing scalability problems in environments featuring large-scale and dynamic-management requirements.

3. The Proposal

We propose QMPSR, a Qualitative Method for Prioritizing Software Requirements. Our approach formalizes the assessment of requirements prioritization and focuses on the project's relevant aspects and elements that define the priority of such requirements. Thus, our proposal allows a high degree of generality to deal with different types of requirements such as functional requirements, security requirement (Vilela, et al., 2017), design restrictions, implementation

requirements, user interface requirements (Sánchez & Macías, 2019; Cayola & Macías, 2018), physical requirements and so on.

The main motivation to create a qualitative method is to take into consideration qualitative issues related to decisions made by participants in the requirements prioritization process. As the literature demonstrates (Schön, et al., 2017; Pitangueira, et al., 2015; Achimugu, et al., 2014), most of the existing proposals do not provide mechanisms to validate the different valuations provided to each requirement during the requirements prioritization process. In this way, the utilization of suitable aspects and qualitative elements (i.e., the requirements' characteristics) allows to get an easy agreement among stakeholders, improving the prioritization process and obtaining more accurate results. Likewise, our motivation is aimed at facilitating the management of requirements, regardless of their type and size, without presenting scalability problems. The objective is to concentrate on defining the qualitative elements that drive the requirements prioritization process. In addition, collision problems, which have never been addressed in previous approaches, provide motivation to carry out our research.

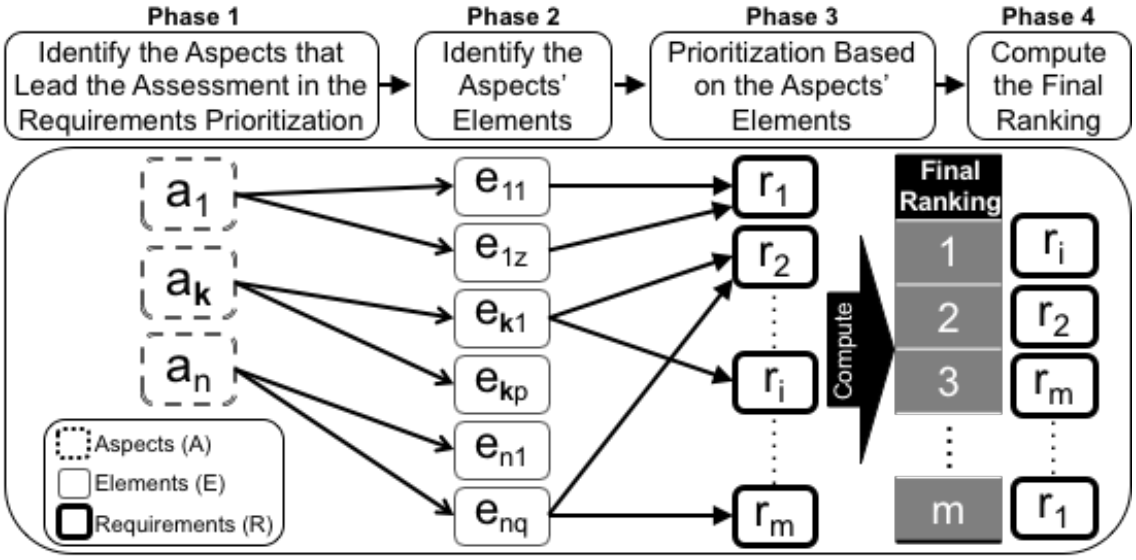

Fig. 1. QMPSR phases.

QMPSR is presented in Fig. 1, including 4 main phases that aim at driving the requirements prioritization process. The project's relevant aspects are described and composed of a set of elements. The requirements prioritization is materialized according to the elements of the project's relevant aspects. The different phases of QMPSR are described below.

3.1. Phase 1: Identify the Project's Relevant Aspects that Lead the Assessment in the Requirements Prioritization Process

The first phase is composed of three steps:

a) **Identify and Define the Project's Relevant Aspects**

The project's relevant aspects correspond to significant issues that drive the assessment in the requirements prioritization. In other words, these are the issues that decision-makers consider when prioritizing requirements that are valuable for the project.

Riegel and Doerr (2015) identified, by means of a systematic review of bibliography, what aspects (prioritization criteria) should be taken into consideration in order to determine the requirements' values. Hence, the model proposed by Riegel and Doerr allows to support identification and definition of the project's relevant aspects. According to that, in order to discern the relevance level of each requirement, software projects should consider mainly, but not exclusively, aspects such as *Usability*, *Strategy-Related Benefit*, and *Business Value,* among others. In a real setting, these aspects are indicated in a generic way and collectively negotiated among decision-makers according to the main guidelines of each project. More specifically, our approach allows to deal with an unlimited number of aspects that can be defined for a concrete project under demand.

b) Define the Priority of the Project's Relevant Aspects

The Priority of the Project's Relevant Aspects is established in terms of the set of aspects identified in the project, generating a ranking of them sorted by relevance. Priorities are collectively assigned among decision-makers in order to identify preferences. Accordingly, for a total of *n* relevant aspects defined in a project, the aspect with lowest relevance is assigned with 1, while the aspect with highest relevance is assigned with *n*.

c) Define the Normalized Priority of the Project's Relevant Aspects

The Normalized Priority of the Project's Relevant Aspects (*P*) is used to represent the priority with regard to the total number of relevant aspects defined in the project. Furthermore, a normalized definition is adopted to guarantee that *P* will take values between 0 and 1, as it is suggested by Botta (2007).

Let $A = \{a_1, \dots, a_k, \dots, a_n\}$ be an ordered finite collection of a project's relevant aspects, where $a_k \in A$, such that *k* represents the priority order of importance, and *n* is the total number of the project's relevant aspects $\{n \in \mathbb{N}: n \geq 1\}$. We define the Normalized Priority of a Project's Relevant Aspect a_k as:

$$P(a_k) = \frac{k}{|A|}, \tag{1}$$

where $P(a_h) > P(a_k)$ means that aspect a_h ($a_h \in A$) has a higher priority than a_k.

3.2. Phase 2: Identify the Elements of the Project's Relevant Aspects

The second phase is composed of two steps:

a) Identify and Define the Elements of the Project's Relevant Aspects

Every project's relevant aspect includes a set of common elements used to drive the valuation for the requirements prioritization. In QMPSR this step consists of identifying and describing those elements for each defined aspect. Taking into consideration the aspects identified by Riegel and Doerr (2015), those may be refined and divided into different subcategories, allowing

to support the identification and definition of specific elements for the project's relevant aspects. For instance, in a software project on electronic commerce some of the elements related to the relevant aspect of the *Business Value* may correspond to, among others, *Sales Service Management, Inventory Management* and *Additional Revenue Streams*. These elements are indicated in a generic way and collectively negotiated among decision-makers. Our approach allows to deal with an unlimited number of elements that can be associated with aspects defined for a concrete project.

b) Define the Priority of the Elements Assigned to each Project's Relevant Aspect

This step consists of identifying the priority of the elements assigned to each aspect defined above. We consider three different priorities: *high, medium* or *low*. This allows to compare the priority among different elements of an aspect, regardless of the number of elements.

Let $E = \{e_{11}, \dots, e_{1z}, \dots, e_{k1}, \dots, e_{kp}, \dots, e_{n1}, \dots, e_{nq}\}$ be a finite collection of elements, where *z, p* and *q* correspond to the total number of elements for aspects a_1, a_k and a_n, respectively, $\{z, p, q \in \mathbb{N}: z, p, q \geq 1\}$. The priority of the element e_{kp} is formally defined in terms of the function $L(e_{kp})$, where $L: E \rightarrow \{1,2,3\}$, each of the values having the following interpretation:

$$L(e_{kp}) = \begin{cases} 1, & \text{if element } e_{kp} \text{ has } low \text{ priority;} \\ 2, & \text{if element } e_{kp} \text{ has } medium \text{ priority;} \\ 3, & \text{if element } e_{kp} \text{ has } high \text{ priority.} \end{cases} \quad (2)$$

3.3. Phase 3: Requirements Prioritization Process Based on Elements of the Project's Relevant Aspects

The third phase in QMPSR aims at obtaining the requirements prioritization through the elements of a project's relevant aspects. In this phase, requirements prioritization is performed by identifying the relationships between the elements of the project's relevant aspects and the requirements. In this case, one element may be associated with one or many different requirements and a requirement can be associated with one, many or no elements of the project's relevant aspects. Such relationships are specified and collectively negotiated among decision-makers in order to represent the assessment of requirements prioritization. For example, a requirement identifying a business asset ready for sale may be associated with the element *Inventory Management* of the aspect *Business Value*. The identification of these relationships allows decision-makers to focus on the discussion of the requirements' priority on qualitative elements (of the project's relevant aspects) and formally argue their relevance (priority).

Let $R = \{r_1, \dots, r_i, \dots, r_m\}$ be a finite collection of requirements, where r_i is a requirement of the project and *m* is the total number of requirements defined $\{m \in \mathbb{N}: m \geq 1\}$. The relationship among the requirement r_i and the element e_{kv} ($e_{kv} \in E$) is formally defined in terms of the function $C(e_{kv}, r_i)$, where $C: ExR \longrightarrow \{0,1\}$, each of the values having the following interpretation:

$$C(e_{kv}, r_i) = \begin{cases} 1, & \text{if element } e_{kv} \text{ is related to req. } r_i; \\ 0, & \text{otherwise.} \end{cases} \quad (3)$$

3.4. Phase 4: Compute the Final Ranking

The last phase in QMPSR aims at generating a final ranking of requirements based on the project's relevant aspects. This phase includes two steps:

a) Compute the Relevance Level of the Requirements by Aspect

This step consists of identifying the relevance level of the requirements according to the project's aspects. The relevance level of a requirement by aspect is expressed in percentage value and grouped by aspect. More specifically, the relevance of a particular requirement with regard to a specific aspect corresponds to the percentage that represents the sum of the priorities of the elements of a project's relevant aspect associated with the requirement plus an Association Factor (*G*). In this manner, *G* corresponds to the ratio between the number of requirements related to the element and the maximum number of requirements that an element of the same relevant aspect may be associated with. Thus, *G* allows to differentiate the relevance level among elements with equal priority depending on the number of associated requirements. As a result, elements with the same priority but with different number of associated requirements have different relevance level.

In this way, the total number of requirements related to the element e_{kv} is defined as:

$$TC(e_{kv}) = \sum_{i=1}^{m} C(e_{kv}, r_i), \tag{4}$$

where $C(e_{kv}, r_i)$ identifies whether or not the element e_{kv} is related to the requirement r_i, computed with the formula shown in (3), and *m* indicates the total number of requirements defined. Hence, the association factor of the element e_{kv} is defined as:

$$G(e_{kv}, e_{kb}) = \frac{TC(e_{kv})}{TC(e_{kb})}, \tag{5}$$

where $TC(e_{kb})$ is the maximum number of requirements that an element of the aspect a_k has been associated with, and $e_{kb} \in E, \forall e_{kx} \in E, TC(e_{kb}) \geq TC(e_{kx})$.

Fig. 2 illustrates the requirements prioritization, also showing the implication of *G* on the Relevance Level of the Requirements by Aspect (λ). Requirements r_1 and r_3 are associated with elements e_{k1} and e_{k2}, respectively, which have equal priority (*high*) but different λ because *G* is different for each element (1 and 0.5, respectively). Thus, a greater number of requirements associated with an element corresponds to a greater *G* and so its priority is higher. It can also be observed that the requirement r_2 gets the highest $\lambda(a_k, r_2)$. This relevance level is justified by the association of elements e_{k1} and e_{k3} with the requirement r_2.

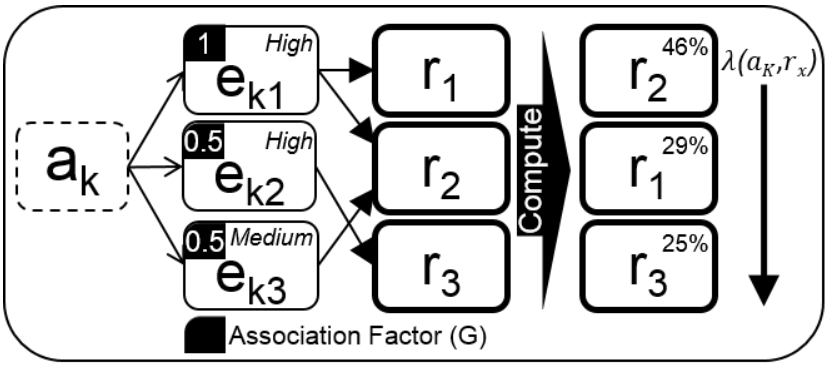

Fig. 2. Influence of the Association Factor (G) on the Relevance Level of the Requirements by Aspect (λ).

The priority of an element e_{kv} of the project's relevant aspect a_k, associated with the requirement r_i, and taking also into consideration its *G*, is formally defined in terms of the function $I: ExR \rightarrow \mathbb{R}$, depicted as follows:

$$I(e_{kv}, r_i) = \begin{cases} L(e_{kv}) + G(e_{kv}, e_{kb}), & \text{if } C(e_{kv}, r_i) = 1; \\ 0, & \text{otherwise.} \end{cases} \quad (6)$$

Where $L(e_{kv})$ defines the priority function for the element e_{kv}, as per (2), while $G(e_{kv}, e_{kb})$ defines the association factor for the element e_{kv}, computed with the formula shown in (5). Moreover, $C(e_{kv}, r_i)$ identifies whether or not the element e_{kv} is related to the requirement r_i, as defined in (3).

Let $E_k = \{e_{k1}, \ldots, e_{kp}\}$ be a finite sub-collection of all elements related to aspect a_k $(E_k \subset E)$, where $|E_k|$ corresponds to the total number of elements ($p = |E_k|$). Hence, the total number of elements of the aspect a_k assigned to the requirement r_i is defined as:

$$TI(a_k, r_i) = \sum_{v=1}^{|E_k|} I(e_{kv}, r_i), \quad (7)$$

where $|E_k|$ indicates the total number of the elements defined for the aspect a_k. Finally, we define the relevance level of the requirement r_i with regard to the aspect a_k in percentage terms as:

$$\lambda(a_k, r_i) = \frac{TI(a_k, r_i)}{\sum_{v=1}^{|E_k|} \left(\left(L(e_{kv}) + G(e_{kv}, e_{kb}) \right) TC(e_{kv}) \right)} 100, \quad (8)$$

where $\lambda(a_k, r_i) > \lambda(a_k, r_j)$ means that r_i has higher priority than r_j for the aspect a_k. For example, Fig. 2 allows to observe the relevance level of the requirements r_1, r_2 and r_3 with regard to the aspect a_k, where these requirements obtained 29%, 46% and 25%, respectively, as a result of the application of *λ*.

b) Compute the Final Ranking of Requirements According to the Relevance Level of each Requirement by Aspect

The Final Ranking of Requirements (*FR*) attempts to classify the requirements considering their relevance level for all aspects defined in the project. *FR* is computed through two sequential

steps: the first step aims at generating the Relevance Ranking of Requirement by Aspect (*W*), and the second step attempts to weight *W* through *P*.

On the one hand, in order to have *W* it is necessary to compute a sorted list of requirements according to λ. In this way, the order of relevance between the requirements r_i and r_j with respect to the aspect a_k is formally defined in terms of the function $M\lambda(a_k, r_i, r_j)$, where $M\lambda: AxRxR \longrightarrow \{0,1\}$, each of the values having the following interpretation:

$$M\lambda(a_k, r_i, r_j) = \begin{cases} 1, & \text{if } \lambda(a_k, r_i) > \lambda(a_k, r_j); \\ 0, & \text{otherwise.} \end{cases} \quad (9)$$

Where $\lambda(a_k, r_i)$ defines the relevance level function for the requirement r_i with regard to the aspect a_k in percentage terms, as defined in (8). In this way, the total number of requirements that have a higher relevance level than requirement r_i with respect to the aspect a_k is defined as:

$$TM\lambda(a_k, r_i) = \sum_{j=1}^{m} M\lambda(a_k, r_i, r_j), \quad (10)$$

where $M\lambda(a_k, r_i, r_j)$ defines the order of relevance for requirements r_i and r_j with respect to the aspect a_k, as described in (9), and *m* represents the total number of requirements. In this way, we formally define the ranking of the requirement r_i for the aspect a_k in terms of the function *W*, where $W: AxR \rightarrow \{1, \ldots, m\}$ and $\{W(a_k, r_i) \in \mathbb{N}: 1 \leq W(a_k, r_i) \leq m\}$, so that *m* is the total number of requirements of the project. This ranking is carried out by applying a set of rules conducted by the Relevance Level of the Requirements by Aspect (λ):

$$W(a_k, r_i) = \begin{cases} 1, & \text{if } \lambda(a_k, r_i) = 0; \\ m - TM\lambda(a_k, r_i), & otherwise. \end{cases} \quad (11)$$

Where $\lambda(a_k, r_i)$ defines the relevance level for the requirement r_i with regard to the relevant aspect a_k in percentage terms, as defined in (8).

On the other hand, $TM\lambda(a_k, r_i)$ identifies the total number of requirements that have a higher relevance level than requirement r_i with respect to a_k, computed with the formula shown in (10).

Finally, we define the final ranking of the requirement r_i taking into consideration all the project's relevant aspects as:

$$FR(r_i) = \sum_{h=1}^{n} W(a_h, r_i) P(a_h), \quad (12)$$

where $P(a_h)$ defines the Normalized Priority function for the Project's Relevant Aspect a_h, as per (1), and *n* is the total number of relevant aspects defined in the project. Finally, $FR(r_i) > FR(r_j)$ means that r_i has higher priority in the final ranking than r_j. Nevertheless, $FR(r_i) = FR(r_j)$ means that there is a requirements collision between r_i and r_j, which will be addressed in detail in Section 5.

In the next section, an example of implementation of QMPSR through an application scenario is presented, in order to provide a better understanding of the phases described above.

4. Application Scenario

The verification of QMPSR has been addressed through the application of the method to a practical example of development. To carry out this task, the phases described in Section 3 will be applied sequentially.

4.1. General Guidelines

The application chosen to carry out the verification of our approach is a web-based learning management system intended for academic use. The objective of this application is to carry out the main processes of enrollment, management and monitoring of university courses. Likewise, the final users of this information system are students, instructors and administrative staff of a given university. In this way, it is required the implementation of a web application with the following features:

- Instructors and administrators can create, edit and consult the courses.
- Instructors and students can deal with the courses in an integrated environment.
- Instructors can create and edit evaluation activities for each course.
- Students can interact with evaluation activities for each course.
- Administrators can configure the profiles of both students and instructors.
- Administrators can manage course enrollment.

According to the general guidelines outlined above, the requirements for this information system are presented in Table 2.

Table 2

Requirements for the system proposed.

R	Requirements
r_1	Create and configure courses included in the university curricula.
r_2	Manage courses in an integrated environment.
r_3	Create and edit assessment activities.
r_4	Configure the students and instructors' profiles.
r_5	Manage the enrollment of students in different courses.
r_6	Link and provide documents in different formats, associated with the courses.
r_7	Consult one course's events and tasks.
r_8	Check the information about the students' progress.
r_9	Allow to review the results of the assessment activities and notify about the required cycles.
r_{10}	Allow to attach and link digital documents with assessment activities.
r_{11}	Create discussion environments for the members of the courses.
r_{12}	Provide a centralized virtual space to store and maintain digital information.
r_{13}	Create virtual classrooms that allow online teaching.
r_{14}	Allow to adapt the information system according to different profiles.

A real work team (decision-makers related to the project) has been formed to implement the QMPSR method, in order to carry out the management of the project requirements. A total of 5 software engineers (M_{age}=38.8 years, *SD*=11.2, range=28-55 years; 80% male) participated in the implementation of QMPSR. Participants had skills in computer science, specifically in topics related to software-project management in agile environments and related tools.

The general guidelines, requirements and prioritization criteria proposed by (Riegel & Doerr, 2015) were the input for decision-makers to work collectively and collaboratively by themselves in the requirements prioritization, carrying out the different phases of QMPSR presented in Section 3.

4.2.1. Phase 1: Identify the Project's Relevant Aspects that Lead the Assessment in the Requirements Prioritization Process

This initial task helps identify the relevant aspects of the project – i.e., aspects have to be defined and ranked by relevance. In addition, standardized priority for each relevant aspect has to be calculated (see Section 3.1). This task is collectively negotiated among decision-makers according to the project's main guidelines and requirements.

Table 3

Relevant aspects of the learning management system.

Aspects (*A*)	Priority	Normalized Priority ($P(a)$)
Usability	3	1
Content	2	0.66
Business Value	1	0.33

The result of this task is presented in Table 3. As it can be seen, *Usability, Content* and *Business Value* aspects have been defined to formally consider the priorities related to end users, the information architecture and the project itself, respectively, during the prioritization process. These aspects have been selected and prioritized due to the type of application domain (academic environment), where these aspects result more relevant than, for example, the cost of implementing the project, the risk of carrying it out or the expected economic benefit. As mentioned above, Riegel & Doerr (2015) provide a set of aspects that can be considered to support this task.

On the other hand, *Usability, Content* and *Business Value* aspects have a priority of 3, 2 and 1, respectively, where 3 (*Usability*) indicates the most important priority and 1 (*Business Value*) the least important one. According to that, these three aspects have a normalized priority (*P*) of 1, 0.66 and 0.33, respectively, as defined in formula (1). Thus, the requirements prioritization process would be mainly driven by the preferences related to the final users of the project (*Usability*). In this way, the output of this task is the prioritized set of relevant aspects of the learning management system, as specified in Table 3.

4.2.2. Phase 2: Identify the Elements of the Project's Relevant Aspects

This task consists in identifying and prioritizing the elements related to the aspects stated in the previous task. This task is also collectively negotiated among decision-makers according to the main guidelines, requirements and relevant aspects of the project provided above. Table 4 shows the resulting elements identified for each relevant aspect of the project.

Table 4

Elements for the relevant aspects of the learning management system.

Aspects (*A*)	Elements (*E*)	Priority	Priority ($L(e_{kp})$)
Usability	Adaptability.	High	3
	Ease of learning.	High	3
	Content accessibility.	High	3
	Reduced cognitive burden.	Medium	2
	Error tolerance.	Low	1
	Ease of remembrance.	High	3
Content	Personal information of students.	Medium	2
	Content of knowledge units.	High	3
	Personal information of instructors.	Low	1
	Communications between students and instructors.	Medium	2
	Preferences of the students.	High	3
	Description of knowledge units.	High	3
	Feedback of evaluation activities.	High	3
	Registration of student activities.	High	3
Business Value	Manage the students' progress.	Medium	2
	Manage the students' enrollment.	High	3
	Monitor the students' learning process.	High	3
	Manage knowledge units.	High	3
	Organize the storage of information.	Medium	2
	Coordinate composition of knowledge units.	High	3

As well as aspects, elements were selected and prioritized according to their relevance for the web-based environment proposed. For instance, for the *Usability* aspect, the element "*Ease of learning*" was defined, with the objective of formally considering the importance of providing an effective interaction with the system. In fact, this element is given a higher priority than the element "*Error tolerance*" (*High* and *Low*, respectively). As mentioned above, the aspects identified by Riegel & Doerr (2015) can be refined into subcategories to support their identification and definition. These elements have been obtained and prioritized according to the requirements and the characteristics of the project. It is worth noting that there is no direct relationship between the proposed prioritization criteria and the aspects and elements defined, because the latter were identified and adjusted based on the specific characteristics of the project. However, it is possible to consider prioritization criteria, identified by Riegel & Doerr (2015), more related to the aspects and elements defined, which correspond to *Benefit and Product / System Quality* categories and their subcategories such as *Business Value, Business importance and Gain for Organization, Ease of Use / Convenience, Scalability, Sustainability of Solution, Changeable Solution, Uniform Solution, Performance, Stability, Security, Integrity, Availability, Testability* and *Accuracy*, respectively.

As shown in Table 4, *Usability, Content* and *Business Value* aspects are composed of 6, 8 and 6 elements, respectively. Likewise, each element is assigned a priority (*L*), using the *High, Medium* or *Low* scale, computed with the formula (2). In this way, the output of this task is the set of elements according to the relevant aspects of the learning management system proposed (Table 4).

4.2.3. Phase 3: Requirements Prioritization Process Based on Elements of the Project's Relevant Aspects

Once the learning management system's aspects and relevant elements have been identified, the next task corresponds to the prioritization of the requirements. This task implies to identify the relationship between the elements of the relevant aspects (see Table 4) and the requirements (see Table 2). These relationships are collectively identified among decision-makers according to the main guidelines, requirements and elements of the relevant aspects of the project.

The output of this task is presented in Table 5. As it can be seen, while the first and second columns depict the aspects and elements, respectively, the following columns represent the project's requirements. In this way, it is possible to identify the requirements (marked with 1) related to each element, as defined in formula (3).

Table 5

Requirements prioritization process based on elements of the project's relevant aspects.

Aspects	Elements	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	R11	R12	R13	R14
Usability	Adaptability.	0	1	0	0	0	0	0	0	0	0	0	0	0	0
	Ease of learning.	0	1	0	0	0	0	0	0	0	0	0	0	0	0
	Content accessibility.	0	0	0	1	0	0	1	1	0	0	0	0	1	1
	Reduce cognitive burden.	0	0	0	0	0	0	1	0	0	0	0	0	0	0
	Error tolerance.	1	0	1	0	0	1	0	0	0	1	0	0	0	0
	Ease of remembrance.	0	1	0	0	0	0	0	0	0	0	0	0	0	0
Content	Students' personal information.	0	0	0	1	0	0	0	0	0	0	0	0	0	0
	Content of knowledge units.	0	0	0	0	0	1	0	0	0	0	0	0	0	0
	Instructors' personal information.	0	0	0	1	0	0	0	0	0	0	0	0	0	0
	Communications between students and instructors.	0	0	0	0	0	0	0	0	0	0	1	0	0	0
	Students' preferences.	1	0	0	0	0	0	0	0	0	0	0	0	0	0
	Description of knowledge units.	1	1	0	0	0	0	0	0	1	0	0	0	0	0
	Feedback of evaluation activities.	0	0	1	0	0	0	0	0	1	0	0	0	0	0
	Registration of student's activities.	0	0	0	0	0	0	0	1	0	1	0	0	0	0
Business Value	Manage the students' progress.	0	1	0	0	0	0	0	0	0	0	0	0	1	0
	Manage the students' enrollment.	0	0	0	0	1	0	0	0	0	0	0	0	0	0
	Monitor the students' learning process.	0	0	1	0	0	0	0	0	1	0	0	0	0	0
	Manage knowledge units.	1	0	0	0	0	0	0	0	0	0	0	0	0	0
	Organize the storage of information.	0	0	0	0	1	0	0	0	0	0	0	1	0	0
	Coordinate composition of knowledge units.	1	0	0	1	1	1	0	0	1	0	0	0	0	0

4.2.4. Phase 4: Compute the Final Ranking

Once the linking with the requirements has been generated among decision-makers, the last task consists of generating the Final Ranking of Requirements (*FR*). This task implies computing automatically the *FR* in two steps, according to the relationships previously established (see Table 5, which is the main input for this task) and the formulas defined in section 3.4.

This first step consists of identifying the relevance level of the requirements according to the project's aspects. In this way, the total number of requirements related to each element is computed with the formula shown in (4). For example, the element "*Content accessibility*" of the *Usability* aspect includes 5 related requirements. This element has the maximum number of requirements that an element of the *Usability* aspect has been associated with. Therefore, its Association Factor (*G*) is 1 according to formula (5). This allows to compute formula (6). For instance, the priority of the previous element, associated with the requirement r_4, is 4 taking into consideration its *G*. However, a complete understanding of the requirements' relevance level also implies considering all the elements associated with each aspect, and comparing the results with all the requirements. The above is obtained by computing the formulas (7) and (8). Thus, the relevance level of each requirement (i.e., from r_1 to r_{14}) with regard to the *Usability* aspect, obtained values of 4.62%, 24.62%, 4.62%, 10.26%, 0%, 4.62%, 15.9%, 10.26%, 0%, 4.62%, 0%, 0%, 10.26% and 10.26%, respectively, as the result of the application of Relevance Level of the Requirements by Aspect (*λ*). In this way, results for requirements r_1 (4.62%) and r_2 (24.62%) indicate that r_2 has higher priority than r_1 for the *Usability* aspect.

The second step is to generate the Relevance Ranking of Requirements by Aspect (*W*). It is intended to weight *W* through *P*. To obtain W, the order of relevance for the requirements related to each aspect is calculated according to formulas (9) and (10). In this way, a set of rules conducted by *λ* is applied to the sorted list of requirements, as defined in (11). According to that, the ranking of the requirements (i.e., from r_1 to r_{14}) for the *Usability* aspect in terms of the function *W* corresponds to: 8, 14, 8, 12, 1, 8, 13, 12, 1, 8, 1, 1, 12 and 12, respectively. Finally, the *FR* is generated, considering the level of relevance for each requirement by aspect and its normalized priority (*P*), computed with the formula (12).

Table **6** presents the *FR* for the requirements of the learning management system proposed. As it can be seen, the requirements are sorted according to their *FR*, calculated with the formula (12). In this way, r_2 and r_{12} are the requirements with the highest and lowest priority, respectively. These requirements obtain a final ranking of 24.56 and 4.3, respectively.

Table 6

Final ranking of requirements for the learning management system proposed.

ID	Requirements	Final Ranking (*FR*)
r_2	Manage courses in an integrated environment.	24.56
r_4	Configure the students and instructors' profiles.	22.89
r_1	Create and configure courses included in the university curricula.	20.54
r_8	Check the information about the students' progress.	19.59

r_3	Create and edit assessment activities.	18.23
r_6	Link and provide documents in different formats, associated with the courses.	16.25
r_{10}	Allow to attach and link digital documents with assessment activities.	15.59
r_{13}	Create virtual classrooms that allow online teaching.	15.3
r_9	Allow to review the results of the assessment activities and notify about the required cycles.	14.53
r_7	Consult one course's events and tasks.	13.99
r_{14}	Allow to adapt the information system according to different profiles.	12.99
r_5	Manage the enrollment of students in different courses.	6.28
r_{11}	Create discussion environments for the members of the courses.	5.29
r_{12}	Provide a centralized virtual space to store and maintain digital information.	4.3

It is worth noting the ability of the QMPSR prioritization method to minimize possible requirements collisions through this application scenario. For example, requirements r_3 and r_6 are associated with elements that have the same priority (see Table 4 and Table 5). That is, these requirements are associated with an element of the *Usability* aspect with *Low* priority, an element of the *Content* aspect with *High* priority, and an element of the *Business Value* aspect with *High* priority. However, requirements r_3 and r_6 obtain different final ranking (*FR*) values (18.23 and 16.25, respectively), as the Association Factor (*G*), computed with the formula (5), is different for each element. It can be observed in Table 5 that the requirements r_3 and r_6 are related to elements "*Feedback of evaluation activities*" and "*Content of knowledge units*", respectively, both of *High* priority and belonging to the *Content* aspect. However, the element "*Feedback of evaluation activities*" has a greater number of associated requirements (see Table 5), generating, as a result, a higher priority. In this way, as already mentioned in Section 3.4, an element with a greater number of associated requirements corresponds to a greater *G* and, consequently, its priority is higher. This allows to reduce collisions between requirements associated with elements with the same priority, generating precise results in the final ranking of requirements.

It is worth mentioning that phases 1 and 2 required the lowest and highest manual effort for decision-makers, respectively, as they had to work collaboratively to concrete aspects and related elements. Due to the dynamic nature of the prioritization process, some effort was also required during the phase 3 to adjust and define elements coming from phase 2.

All in all, this application scenario is only an example to illustrate the method. In the next section, we provide experiments, including higher sets of requirements, to evaluate QMPSR against other existing prioritization methods and thus demonstrate the advantages of our approach.

5. Comparative Framework for the Empirical Evaluation of Prioritization Methods

The challenge of assessing different prioritization methods can be faced in different ways. One of the most common solutions is setting up an empirical case scenario with synthetic data and virtual subjects (Achimugu, et al., 2014; Greer & Ruhe, 2004; Wiegers, 1999; Kaiya, et al., 2002). This approach allows to appreciate the features of every proposal lacking, however, implementation in a real-world project and comparative assessment of its features with other

prioritization methods. Another approach is to conduct the evaluation of prioritization methods through experiments with real subjects by setting case studies in real-world projects (Karlsson & Ryan, 1997; Logue & McDaid, 2008; Azar, et al., 2007). This would enable to review the results on the characteristics of the proposals. However, one of the main problems with this approach *is the shortfall of ground truth* (called *true ranking* in requirements prioritization), which leads to the lack of a quantitative measure of the quality concerning the resulting priority rank (Perini, et al., 2013).

It is worth noting that the assessment of prioritization methods is usually focused on reviewing the results of the features without conducting a comparative analysis with other proposals. Although there are specific researches that focus on achieving such comparative assessments (Karlsson, et al., 1998; Alshehri & Benedicenti, 2013; Ahl, 2005), these validations put emphasis exclusively on the analysis of the users' feedback through the utilization of subjective measures (e.g., *ease of use, reliability of results* or *fault tolerance*). Some other approaches are based on objective measures, but exclusively focused on the required number of decisions or time elapsed (Berander, et al., 2006).

Perini et al. (2013) proposed simulations combining synthetic data and a case study with stakeholders (real-world project) in order to assess the CBRank prioritization method. This was indeed achieved using the AHP method, featuring both methods' (CBRank and AHP) similar characteristics (Avesani, et al., 2003). This facilitates carrying out comparative evaluations without identifying similarities or connections with other prioritization methods.

Given the discussion in this section, it could be stated that there are no proposals on prioritization methods that evaluate the performance of a particular feature through a comparative analysis of its results with different methods under similar conditions in order to discover relationships, differences or similarities in the studied approaches. The heterogeneity of the prioritization methods, with regard to the type of configuration required for both implementation and evaluation, makes it difficult to contrast and evaluate the same feature in different proposals. One approach to address and drive the abovementioned drawback is to define a comparative framework in order to determine similarities and common patterns among prioritization methods. In fact, it is essential to compose frameworks to accomplish experiments with different proposals under similar conditions.

Taking up such challenge, we have created a framework to carry out the comparative analysis of prioritization methods based on specific features. This also allows to provide evidence about the validation of the prioritization methods (Daneva, et al., 2014). In this way, a number of prioritization methods have been selected and evaluated in our comparative framework in order to observe and compare requirements collisions generated by each of them. Although this measurement criterion (requirements collisions) has not been considered in previous research studies (Daneva, et al., 2014; Berander, et al., 2006), it becomes relevant in different issues of the requirements prioritization process. For example, a large number of requirements collisions increases the time for decision-making and reduces the ability to discern the relevance level of the requirements. On the other hand, a moderate number of requirements collisions allows to

minimize scalability problems, ensures the consistency of the assessments and facilitates the accuracy of the ranking as the number of requirements increases.

Specifically, we have considered the following prioritization methods, all presented in Section 2, in order to analyze collisions: QMPSR (our approach), MoSCoW, Value-Oriented, Wiegers, Product Definition, AHP and Kano. These methods have been considered mainly due to their popularity, their representativeness with respect to other methods and their facilities to fit the guidelines defined for the comparative framework. In fact, we defined the following general guidelines:

Common Definition of Prioritization Effort: prioritization methods use different conceptions and judgments on how the requirements prioritization should be accomplished. Consequently, the first step required to obtain a comparative framework is the creation of a common definition of what is understood as prioritization effort. Firesmith (2004) argues that prioritization dimensions are the factors that influence the priority of a requirement. According to Berander and Andrews (2005), requirements can be prioritized along many different, related and even opposite prioritization dimensions (e.g., *importance, business value, penalty, cost, time,* and *risk*). According to that, we consider prioritization effort as the establishment of the relevance level (e.g., a numerical scale between 0 and 10) of a requirement in a particular prioritization dimension (e.g., *business value)*. It should be noted that the prioritization dimensions in QMPSR correspond to the project's relevant aspects (A), and the establishment of the relevance level is carried out through the association between elements and aspects of a given requirement – i.e., $C(e_{kv}, r_i) = 1$, where *v* corresponds to the total number of elements for aspect a_k and r_i is a requirement of the project. This is what we call *NPD* (Number of Prioritization Dimensions) in our approach, which can be defined as follows: $NPD = \{1, \dots, \iota\}$, i.e., a finite collection of prioritization dimensions, such that ι is the total number of the prioritization dimensions $\{\iota \in \mathbb{N} : \iota \geq 1\}$.

Coordination of Prioritization Effort: together with getting a common understanding of what priority effort is, it is also required to accomplish different experiments with methods using the same level of prioritization effort in order to obtain comparable results. Therefore, the second step needed to compose a comparative framework consists of having a coordinated generation of the prioritization effort between different methods.

Definition of Prioritization Effort Levels: in the last step needed to compose a comparative framework, different Prioritization Effort Levels (*PEL*) are defined with the aim of accomplishing experiments with prioritization methods having different performance conditions. *PEL* value allows to identify and select the number of the prioritization dimensions that will be used to prioritize requirements. In our framework, *PEL* can be defined as follows: $PEL = \{random, minimum, maximum\}$, i.e., a finite collection of prioritization dimensions. For a *random PEL*, the *NPD* for each of the requirements is randomly identified (between 1 and the total *NPD* value defined for each experiment). For a *minimum PEL*, only one of the prioritization dimensions defined for each experiment is randomly identified. Finally, for a *maximum PEL*, all prioritization dimensions defined are selected for each requirement.

These general guidelines allow to arrange different experimental scenarios for each prioritization method in a coordinated manner. Fig. 3 presents the different arrangements that our comparative framework is able to configure.

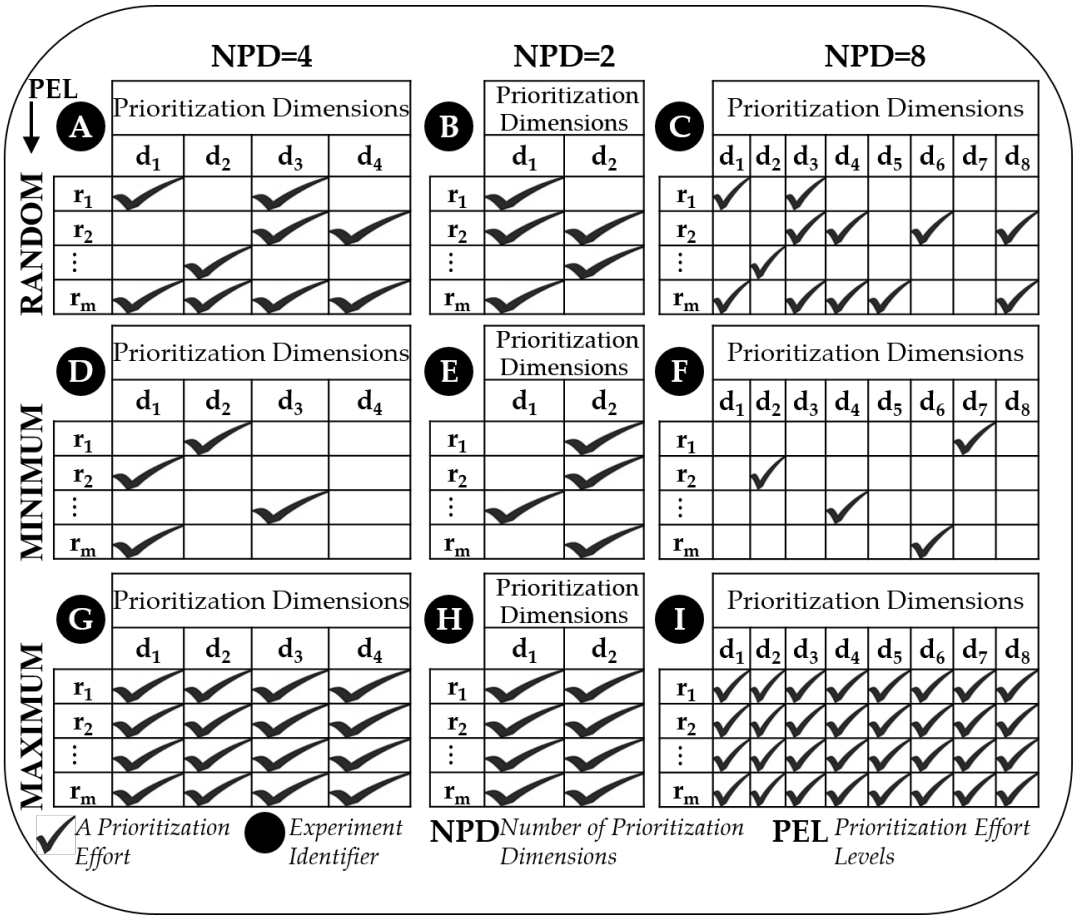

Fig. 3. Different arrangements for the experiments in terms of *PEL* and *NPD*.

As we can see in Fig. 3, 9 different experimental scenarios have been created, varying the *PEL* (*random, minimum* and *maximum*) and the *NPD* (2, 4 and 8). Thus, each experiment is identified by the *PEL* and *NPD* used together with a unique identifier (a capital letter between A and I). Examples of *random PEL* can be observed in experiments A, B and C appearing in Fig. 3, where 4, 2 and 8 prioritization dimensions, respectively, are defined. Similarly, examples of *minimum PEL* can be observed in experiments D, E and F appearing in Fig. 3, where 4, 2 and 8 prioritization dimensions, respectively, are defined. Finally, experiments G, H and I, appearing in Fig. 3, present examples of *maximum PEL*, using 4, 2 and 8 prioritization dimensions, respectively. These experiments will be explained in detail in the next section, featuring formal terminology (Wohlin, et al., 2012).

6. An Experiment on Collided Requirements

As previously mentioned, a requirements collision corresponds to the situation where two or more requirements have the same prioritization value in a final ranking. A large number of collided requirements negatively affects the requirements prioritization process, as it does not

allow to establish differences to prioritize accordingly. Therefore, few collided requirements provide a discernible and accurate ranking. We can identify collided requirements as follows:

Given a final ranking for a set of requirements *R*, called *FR* as defined in (12), the collision of requirement r_i (where $r_i \in R$) is formally defined in terms of the function $K(r_i)$, where $K: R \rightarrow \{0,1\}$, each of the values having the following interpretation:

$$K(r_i) = \begin{cases} 1, & \text{if } \exists r_j \neq r_i, r_j \in R, FR(r_j) = FR(r_i); \\ 0, & \text{otherwise.} \end{cases} \quad (13)$$

In this way, the total number of collided requirements in a particular final ranking is defined as:

$$TK = \sum_{i=1}^{m} K(r_i), \quad (14)$$

where $K(r_i)$ defines the collision function for the requirement r_i, as described in (13), and *m* represents the total number of requirements.

6.1. Configuration Parameters for the Assessment of the Prioritization Methods

Table 7 identifies some characteristics of the selected prioritization methods to be used through the experiments (Sections 6.2 and 6.3, respectively).

Table 7

Description of the Methods. Summary of the main characteristics of the prioritization methods.

Methods	Prioritization Dimensions	Size	Maximum NPD	Type of Evaluation of the Prioritization Dimension
QMPSR	Aspects	Adaptable	unlimited	Elements of each Aspects
MoSCoW	Stakeholders	Adaptable	unlimited	Must, should, could, won't
Value-Oriented	Core business	Adaptable	unlimited	0-10
Wiegers	Benefit, penalty, cost, risk	Semi-adaptable	4	0-9
Product Definition	Technical, creative, user, business	Semi-adaptable	4	0-5
AHP	Requirements	Semi-adaptable	Total number of requirements	1-9
Kano	Present, absent	Non-adaptable	2	Like, expect, don 't care, live with, dislike

The *Prioritization Dimensions* column in Table 7 is used to identify the prioritization dimensions that each method features. As we can see, prioritization dimensions such as *aspects, stakeholders, core business* and *requirements* correspond to those used in QMPSR, MoSCoW, Value-Oriented and AHP methods, respectively. Similarly, *benefit, penalty, cost* and *risk* correspond to the 4 prioritization dimensions used in the Wiegers method, while *technical, creative, user* and *business* correspond to the 4 prioritization dimensions used in the Product Definition method. Finally, *present* and *absent* correspond to the 2 prioritization dimensions used in the Kano method.

The *Size* column in Table 7 is used to identify whether the methods allow to adapt the *NPD* to prioritize each requirement. The *adaptable* value indicates that the method allows to adapt the *NPD*, whereas the *semi-adaptable* value illustrates that the method defines a maximum *NPD* that can be adapted for a specific use. Likewise, the *non-adaptable* value indicates that the method defines a fixed *NPD* to prioritize each requirement. As shown in Table 7, QMPSR, MoSCoW and Value-Oriented methods allow to adapt the *NPD*. On the other hand, Wiegers, Product Definition and AHP methods define a maximum *NPD* (e.g., the Wiegers method defines 4 prioritization dimensions – i.e., *benefit, penalty, cost* and *risk*). However, they allow to adapt the *NPD* value. In the case of AHP, it allows the application of the local stopping rule (Harker, 1987), facilitating to adapt pairwise comparisons of requirements. Finally, Kano method defines a fixed *NPD* to prioritize each requirement.

The *Maximum NPD* column in Table 7 is used to identify whether the methods define a maximum *NPD*. As shown in Table 7, QMPSR, MoSCoW and Value-Oriented methods do not define a maximum *NPD* (*unlimited*). In contrast, the set of prioritized requirements corresponds to the maximum *NPD* defined in the AHP method (the *total number of requirements* is used as the local stopping rule, in order to identify an acceptable error threshold). Finally, the maximum *NPD* defined in Wiegers and in Product Definition methods corresponds to 4, while in Kano method it corresponds to 2.

Finally, the *Type of Evaluation of the Prioritization Dimension* column in Table 7 is used to identify how the requirements prioritization is carried out in each method. As shown in Table 7, Value-Oriented, Wiegers, Product Definition and AHP methods achieve the requirements prioritization by allocating a numerical scale in each prioritization dimension (e.g., the Value-Oriented method uses a numerical scale between 0 and 10 to prioritize each *core business* prioritization dimension). On the other hand, Moscow and Kano methods carry out the requirements prioritization by considering a qualitative scale in each prioritization dimension (e.g., the MoSCoW method uses the scale *must, should, could*, and *won't* to prioritize each *stakeholder*). Finally, our approach (QMPSR) accomplishes the prioritization through the association between elements and *aspects* (prioritization dimensions) for each requirement, as explained in Section 3.

6.2. Research Questions and Experimental Development

In order to follow Design Science, our evaluation and validation have been inspired by the guidelines reported by Hevner et al. (2004) and Wieringa (2009). Our research is based on a problem-driven investigation focused on both goal and impact. In this way, we want to diagnose collision problems in other requirements prioritization algorithms and investigate the impact of the realized implementation (i.e., our approach). For this reason, we have tested our approach through an experimental evaluation, using simulations with synthetic data.

A set of research questions are stated to conduct the work. These will be answered through the results obtained:

- *RQ1*: To what extent does the newly proposed method QMPSR reduce the amount of requirements collisions?
- *RQ2*: To what extent does QMPSR outperform other prioritization methods in terms of requirements collisions?
- *RQ3*: Can QMPSR overcome scalability problems as the number of input requirements increases?

A set of 9 experiments was accomplished to assess the prioritization methods in order to investigate the collisions generated under different performance conditions.

Independent variables considered for the experiments were: *Prioritization Method* (QMPSR, MoSCoW, Value-Oriented, Wiegers, Product Definition, AHP, Kano), *PEL* (*random*, *minimum* or *maximum*), *NPD* (2, 4, 8) and the input sets of synthetic requirements (25, 50, 75, 100, 125, 150, 175 and 200). On the other hand, the dependent variable was the *number of collided requirements*.

Table 8

Experiment identifiers and the methods evaluated in each experiment.

Methods	Experiment Identifier								
	A	**B**	**C**	**D**	**E**	**F**	**G**	**H**	**I**
QMPSR	x	x	x	x	x	x	x	x	x
Value-Oriented	x	x	x	x	x	x	x	x	x
MoSCoW	x	x	x	x	x	x	x	x	x
Wiegers	x	x		x	x		x	x	
Product Definition	x	x		x	x		x	x	
Kano								x	
AHP	x		x				x		x

Table 8 depicts the methods evaluated in each experiment. QMPSR, MoSCoW and Value-Oriented methods were evaluated in all experiments due to their capability to adapt the *NPD* (see the *Size* column in Table 7). On the other hand, Wiegers and Product Definition methods were only evaluated in experiments involving an *NPD* of 2 or 4 – i.e., these methods were not evaluated in experiments with an *NPD* higher than its *Maximum NPD* (see the *Maximum NPD* column in Table 7). Likewise, the AHP method was only evaluated in experiments A, C, G, H and I as it allows to semi-adapt the *NPD* to use (see the *Size* column in Table 7). However, we evaluated AHP in experiments where the total prioritization effort, computed with the formula shown in (16), was twice higher than the number of prioritized requirements. Finally, the Kano method was only evaluated in experiment H, as this method defines a fixed *NPD* (*present* and *absent*) to prioritize each requirement (see the *Prioritization Dimensions* column in Table 7).

6.3. Experiments Execution

Experiments were accomplished applying the Experiment Execution Process illustrated in Fig. 4. The Experiment Execution Process was carried out through the execution of different algorithms. These algorithms were created according to the general guidelines of the comparative framework and the configuration parameters for each method. This means that

there was no interaction with people in the requirements specification process or in the prioritization.

The input data for each experiment were: a value for *PEL* (*random*, *minimum* or *maximum*), a value for *NPD* (2, 4 or 8) and sets of synthetic requirements (25, 50, 75, 100, 125, 150, 175 and 200). Synthetic data have been used in the specification of requirements, without considering a level of abstraction or domain as commonly recommended (Lauesen, 2002) and used in practice, in order to concentrate on the performance of the methods analyzed. The output was the final ranking of requirements for each prioritization method from which we obtain the number of collided requirements. For example, every method involved in experiment A (see Table 8) performed the Experiment Execution Process with *PEL=random* and *NPD=4* for every set of requirements. Steps followed in the Experiment Execution Process are described below (see Fig. 4):

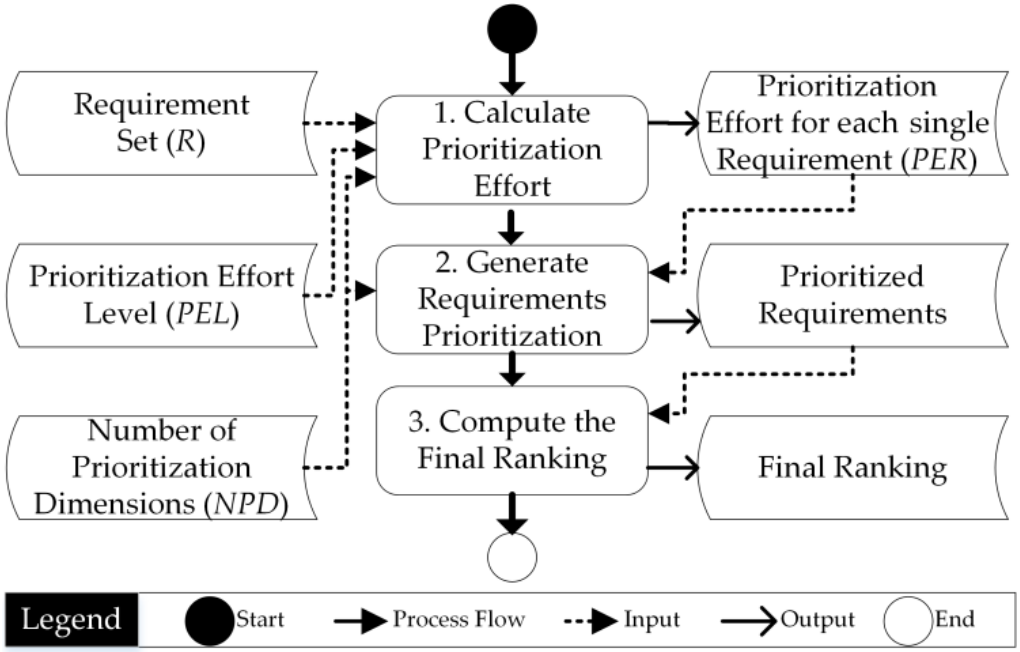

Fig. 4. Experiment Execution Process.

1. Calculate Prioritization Effort: the first step of the process aims at establishing, for every set of requirements, the Prioritization Effort for each single Requirement (*PER*), in order to produce the same level of prioritization effort in all evaluated methods, obtaining comparable results for each of them.

For each set of requirements, *PER* is identified in accordance with *PEL* and *NPD*, both defined for each experiment (see Fig. 3). In this way, given an experiment that considers a particular *PEL* and an *NPD*, the prioritization effort for the requirement r_i is defined formally in terms of the function $PER: RxNPDxPEL \rightarrow \mathbb{N}$ as:

$$PER(r_i, NPD, PEL) = \begin{cases} \beta(1, NPD), \text{if } PEL \text{ is } random; \\ 1, \text{if } PEL \text{ is } minimum; \\ NPD, \text{if } PEL \text{ is } maximum. \end{cases} \quad (15)$$

Where $\beta(1, NPD)$ is a random number in the interval (κ, η) ($\kappa \in Z$ and $\eta \in Q$), formally defined in terms of the function β, where $\beta: ZxQ \rightarrow \mathbb{N}$ and $\{\kappa, \eta \in \mathbb{N}: \kappa \leq \eta\}$, and being *Z* and *Q* collections of natural numbers ($Z, Q \subset \mathbb{N}$). More specifically, in this case β identifies a random number ranged between 1 and *NPD*. Hence, $PER(r_i, NPD, PEL) > PER(r_j, NPD, PEL)$ means that r_i involves a higher number of selected prioritization dimensions than r_j to determine its relevance level. It is worth noting that the *PER* value is the same in all experiments with a *maximum* or *minimum PEL* – i.e., *PER=PEL* in experiments with a *maximum PEL*, while *PER=1* in experiments with a *minimum PEL*. By way of example, in Fig. 3 the prioritization effort for requirements r_1 and r_2 in experiment C (where *PEL=random* and *NPD*=8) is 2 and 4, respectively, as a result of the application of *PER* on both requirements.

In this way, given an experiment, the total prioritization effort for a given set of requirements is defined as:

$$TPER(NPD, PEL) = \sum_{i=1}^{m} PER(r_i, NPD, PEL), \tag{16}$$

where $PER(r_i, NPD, PEL)$ defines the prioritization effort function for the requirement r_i, as defined in (15), and *m* represents the total number of requirements included in a specific set.

2. Generate Requirements Prioritization: the second step in the process aims at establishing the requirements prioritization for each method.

First, a *PER* random number, as defined in (15), is calculated for each requirement. Let $PD = \{d_1, \dots, d_k, \dots, d_{NPD}\}$ be a finite collection of prioritization dimensions, where $d_k \in PD$, such that *NPD* is the total number of prioritization dimensions defined in the experiment. The relationship between the requirement r_i and the prioritization dimension d_k is formally defined in terms of the function $\phi(d_k, r_i)$, where $\phi: PD\ x\ R \longrightarrow \{0,1\}$, each of the values having the following interpretation:

$$\phi(d_k, r_i) = \begin{cases} 1, & \text{if } d_k \text{ is selected for } r_i; \\ 0, & \text{otherwise.} \end{cases} \tag{17}$$

In this way, given an experiment that considers a particular *PEL* and an *NPD*, the total number of prioritization dimensions randomly selected for the requirement r_i ($T\phi$) is defined as:

$$T\phi(r_i, NPD, PEL) = \begin{cases} \sum_{h=1}^{NPD} \phi(d_h, r_i), & \text{if } PEL \text{ is } random; \\ 1, & \text{if } PEL \text{ is } minimum; \\ NPD, & \text{if } PEL \text{ is } maximum. \end{cases} \tag{18}$$

Where $\phi(d_h, r_i)$ identifies whether or not the prioritization dimension d_h is selected for the requirement r_i, computed with the formula shown in (17). Furthermore, the function $T\phi(r_i, NPD, PEL)$ also has the following property $\mathrm{T\phi}(\mathrm{r_i}, NPD, PEL) = \mathrm{PER}(\mathrm{r_i}, NPD, PEL)$. Hence, $\forall (r_i, r_j) \in R, T\phi(r_i, NPD, PEL) > T\phi(r_j, NPD, PEL)$ means that the requirement r_i has a higher number of selected prioritization dimensions than requirement r_j.

Secondly, the requirements prioritization is carried out by assigning an evaluation in each selected prioritization dimension for each requirement. Thus, the prioritization is randomly accomplished in accordance with the evaluation type of each method (see the Type of Evaluation of the Prioritization Dimension column of Table 7).

Initially, the evaluation type for each method needs to be represented in order to achieve the requirements prioritization. Let $S = \{l_1, \dots, l_q, \dots, l_h\}$ be a finite collection of rating-scale values, where $l_q \in S$, such that *h* is the total number of the rating-scale values considered. The evaluation of the requirement r_i in the prioritization dimension d_k, using the rating scale *S,* is formally defined in terms of the function $\psi(d_k, r_i)$, where $\psi: PD\ x\ R \longrightarrow S$, defined as follows:

$$\psi(d_k, r_i) = \begin{cases} l_{\beta(1,|S|)}, & \text{if } \phi(d_k, r_i) = 1; \\ 0, & \text{otherwise.} \end{cases} \quad (19)$$

Where $\beta(1, |S|)$ identifies a random number ranged between 1 and the total number of the rating-scale values considered ($|S|$), $l_{\beta(1,|S|)} \in S$, and $\phi(d_k, r_i)$ identifies whether or not the prioritization dimension d_k is selected for the requirement r_i, computed with the formula shown in (17). Thus, if the prioritization dimension d_k is selected for the requirement r_i – i.e., $\phi(d_k, r_i) = 1$, then the prioritization of the requirement r_i in the prioritization dimension d_k is carried out by randomly assigning a rating-scale value ($l_{\beta(1,|S|)}$). Otherwise, the prioritization of the requirement r_i in the prioritization dimension d_k is not carried out because the prioritization dimension d_k is not selected for the requirement r_i – i.e., $\phi(d_k, r_i) = 0$.

For instance, as shown in Table 7, for the Value-Oriented method a numerical scale between 0 and 10 – i.e., $S = \{0 \dots 10\}$, will be randomly used for each selected prioritization dimension (in this case: *core business*) in order to prioritize each requirement. In QMPSR, by contrast, prioritization is carried out through the association between the requirement's elements and aspects. In this way, QMPSR requires the following steps to prioritize each requirement:

a) Identification of the Number of Elements for each Selected Aspect: In this case, *PD=A*, thus for each aspect d_k ($d_k \in A$), the number of elements associated with the requirement r_i is randomly arranged. The number of elements associated with aspect d_k for the requirement r_i is formally defined in terms of the function $EN(r_i, d_k)$, where $EN: RxA \rightarrow \mathbb{N}$, defined as follows:

$$EN(r_i, d_k) = \begin{cases} \beta(1, |E_k|), & \text{if } \phi(d_k, r_i) = 1; \\ 0, & \text{otherwise.} \end{cases} \quad (20)$$

Where $\beta(1, |E_k|)$ identifies a random number ranged between 1 and the total number of elements of the aspect d_k ($|E_k|$), and $\phi(d_k, r_i)$ identifies whether or not the aspect d_k is selected for the requirement r_i, as defined in (17). Thus, if the aspect d_k is selected for the requirement r_i – i.e., $\phi(d_k, r_i) = 1$, then a random number of elements for the aspect d_k – i.e., $\beta(1, |E_k|)$ is identified for the requirement r_i. Otherwise, no element for the aspect d_k is identified for the requirement r_i. Hence, $EN(r_i, d_k) > EN(r_i, d_h)$ means that

the aspect d_k has a higher number of elements associated with the requirement r_i than the aspect d_h (where $d_h \in A$).

b) Association between the Elements of a Selected Aspect and the Requirements: A random number of *EN*, as defined in (20), associated with aspect d_k is calculated for the requirement r_i. The total number of elements of the aspect d_k randomly associated with the requirement r_i is defined formally in terms of the function $T\tau(r_i, d_k)$, where $T\tau: RxA \longrightarrow \mathbb{N}$, as follows:

$$T\tau(r_i, d_k) = \sum_{v=1}^{|E_k|} C(e_{kv}, r_i), \tag{21}$$

where $C(e_{kv}, r_i)$ identifies whether or not the element e_{kv} is related to the requirement r_i, as defined in (3), and E_k is a finite sub-collection of all elements related to aspect a_k ($E_k \subset E$). Thus $|E_k|$ identifies the total number of elements of the aspect d_k. The function $T\tau(r_i, d_k)$ also has the following property: $T\tau(r_i, d_k) = EN(r_i, d_k)$. Finally, $T\tau(r_i, d_k) > T\tau(r_j, d_k)$ means that requirement r_i has a higher number of associated elements for the aspect d_k than requirement r_j.

3. Compute the Final Ranking: the final step in the experiment execution is to compute the final ranking of requirements for each prioritization method. Each method applies its own procedure. For instance, QMPSR computes the final ranking of requirements according to the formula shown in (12).

In general, each of the 7 methods involved was executed performing the Experiment Execution Process 10 times with the same data set. As a result, the methods included in experiment A (see the first two columns in Table 8) performed the Experiment Execution Process 80 times (10 executions for each of the 8 sets of requirements, with a *PEL=random* and *NPD*=4). Thus, experiment A generated a total number of 480 executions of the Experiment Execution Process for the 6 prioritization methods considered.

6.3.1 Synthetic Data

We utilized two different kinds of synthetic data. The first one corresponds to synthetic data that do not vary through the different experiments and have a general specification. This is the case for the sets of requirements (25, 50, 75, 100, 125, 150, 175 and 200), which do not include any abstraction or domain information in its specification. They were used in all the experiments. This is also the case for the Number of Prioritization Dimension (NPD) for each requirement in experiments where there was a maximum Prioritization Effort Level (PEL).

On the other hand, we also utilized synthetic data generated in a random manner, according to a previously defined set of possibilities (specified in Table 7). For example, the Prioritization Effort for each single Requirement (PER) was randomly defined according to the Number of Prioritization Dimension (NPD) for each experiment. Another example is the Type of Evaluation of the Prioritization Dimension, which was randomly assigned to each selected requirement. It is worth noting that the synthetic data generated in a random way can imply certain variability in

the behavior / performance of the methods. However, these data were also necessary to produce scenarios with different conditions to evaluate the prioritization methods.

In this way, different actions were implemented in order to address the sensitivity to input data. In the first place, all the instances susceptible of being generated in a random way were identified, so that the results could not be determined in advance in any case. Secondly, random results were recorded in order to be replicated for all methods. For example, the Prioritization Effort randomly identified for each single requirement was the same for all the methods involved. Finally, the results obtained correspond to an average of 10 executions for each set of requirements. All this helps reduce the variability in the results.

6.4 Results and Discussion

Table 9 reports the total prioritization effort, computed with the formula shown in (16), generated in each experiment for sets of 25, 50, 75, 100, 125, 150, 175 and 200 requirements. The total prioritization effort represents the number of evaluations carried out for each set of input requirements in order to determine their relevance level.

Table 9

Prioritization Effort.

Experi-ments	Number of Prioritized Requirements							
	25	**50**	**75**	**100**	**125**	**150**	**175**	**200**
A	65	129	191	246	311	374	435	501
B	38	74	112	149	191	223	264	300
C	112	220	336	458	551	672	781	898
D	25	50	75	100	125	150	175	200
E	25	50	75	100	125	150	175	200
F	25	50	75	100	125	150	175	200
G	100	200	300	400	500	600	700	800
H	50	100	150	200	250	300	350	400
I	200	400	600	800	1000	1200	1400	1600

As a result of the aforementioned executions, collided requirements can be represented and analyzed.

On the one hand, Fig. 5, 6 and 7 represent the results obtained from experiments A, B and C, respectively. Each graph shows the number of collided requirements (y-axis), computed via (14), for 25, 50, 75, 100, 125, 150, 175 and 200 prioritized requirements (x-axis). The collided requirements were calculated as an average over 10 executions considering a *random PEL* and prioritization dimensions configured with different sizes (4, 2 and 8 prioritization dimensions in experiments A, B and C, respectively). In addition, and for the case of our approach (QMPSR), we included SD error bars for each execution in order to observe the obtained dispersion. Besides, Mann-Whitney-Wilcoxon test was used to evaluate the difference of the means among the number of collided requirements obtained by QMPSR and the other methods in experiments A, B and C. In all calculations, the *p-value* was < 0.05, indicating that the differences obtained were statistically significant.

On the other hand, Fig. 8, 9 and 10 present the results obtained from experiments D, E and F, respectively. Each graph shows the number of collided requirements (y-axis), computed via (14), for 25, 50, 75, 100, 125, 150, 175 and 200 prioritized requirements (x-axis). The number of collided requirements was calculated as an average over 10 executions considering a *minimum PEL* and prioritization dimensions configured with different sizes (4, 2 and 8 prioritization dimensions in experiments D, E and F, respectively). In these experiments, Mann-Whitney-Wilcoxon test was not calculated because they produced a similar number of collided requirements for each set of requirements. Furthermore, and for the case of our approach, we incorporated SD error bars for each execution with the purpose of examining the dispersion.

Finally, Fig. 11, 12 and 13 illustrate the results obtained from experiments G, H and I, respectively. Each graph presents the number of collided requirements (y-axis), calculated via (14), for 25, 50, 75, 100, 125, 150, 175 and 200 prioritized requirements (x-axis). The number of collided requirements was computed as an average over 10 executions considering a *maximum PEL* and prioritization dimensions configured with different sizes (4, 2 and 8 prioritization dimensions in experiments G, H and I, respectively). Moreover, and for the case of QMPSR, we incorporated SD error bars for each execution with the aim of examining the dispersion. In addition, Mann-Whitney-Wilcoxon test was used to evaluate the difference of the means among the number of collided requirements obtained by QMPSR and the other methods in experiments G, H and I. In all calculations, the *p-value* was < 0.05, indicating that the differences obtained were statistically significant.

6.5 Discussion

In this section, the results of the 9 experiments are analyzed and discussed in order to find answers to research questions *RQ1*, *RQ2* and *RQ3*.

Experiments A, B and C provide evidence of the collisions generated by each prioritization method with a *random PEL* and an *NPD* of 4, 2 and 8, respectively. In all cases, QMPSR outperforms all compared methods.

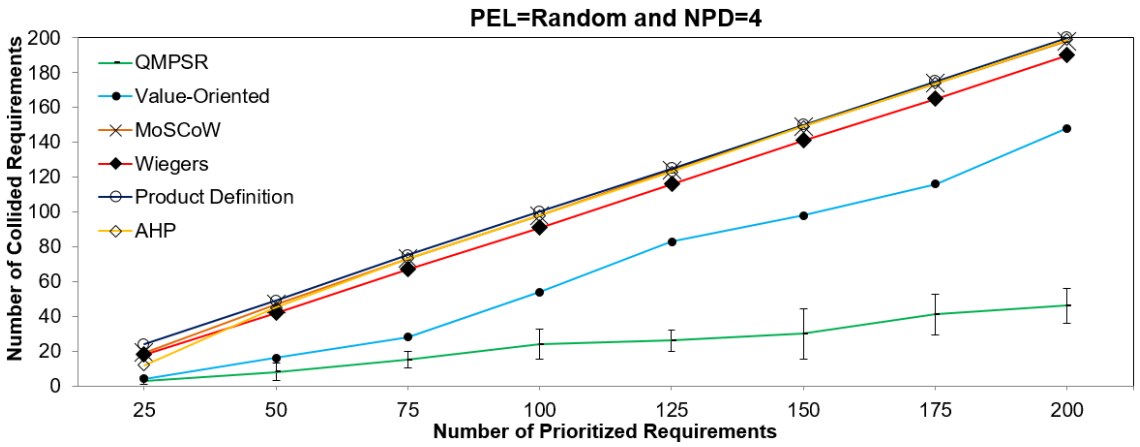

Fig. 5. Experiment A. Number of collided requirements (y axis) for all methods evaluated: QMPSR, Value-Oriented, Moscow, Wiegers, Product Definition and AHP. Values obtained as an average over 10 executions for 25, 50, 75, 100, 125, 150, 175 and 200 prioritized

requirements (x axis), considering *PEL=random* and *NPD*=4. Error bars (SD) are included for our approach (QMPSR) to show the dispersion.

On the one hand, results obtained from experiment A (see Fig. 5) show that QMPSR generates less collided requirements for all sets of input requirements. Even when the Value-Oriented method obtains the second-best performance (- i.e., QMPSR and Value-Oriented methods obtain an average of 19.9% (SD=3.82) and 51.41% (SD=19.31) collided requirements, respectively), the difference with respect to the QMPSR increases when the number of input requirements grows.

On the other hand, experiment B (see Fig. 6) also provides evidence of less collided requirements for all sets of requirements. Only for sets of 25 and 50 requirements the Value-Oriented method gets similar results to QMPSR. Here the difference of collided requirements between Value-Oriented and QMPSR methods is less than 12%, while for all other sets of requirements the number of collided requirements is greater than 35%.

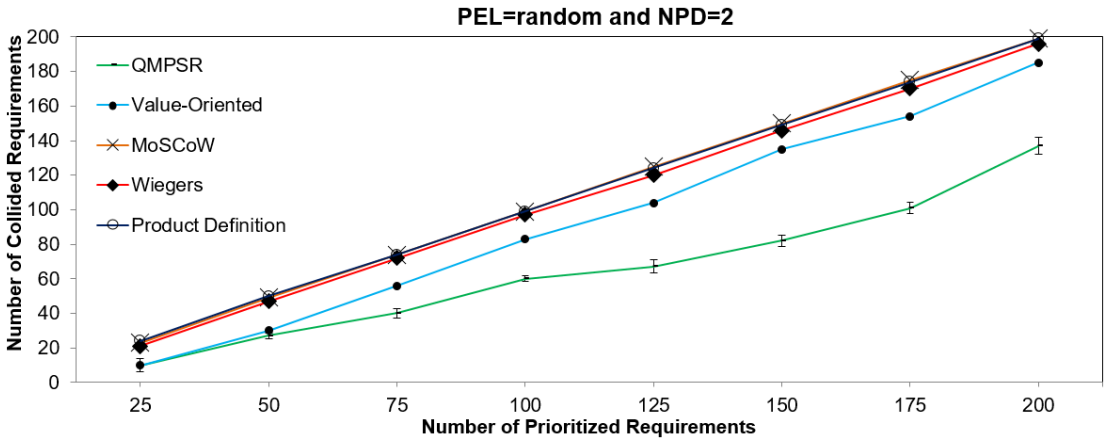

Fig. 6. Experiment B. Number of collided requirements (y axis) for all methods evaluated: QMPSR, Value-Oriented, Moscow, Wiegers and Product Definition. Values obtained as an average over 10 executions for 25, 50, 75, 100, 125, 150, 175 and 200 prioritized requirements (x axis), considering *PEL=random* and *NPD*=2. Error bars (SD) are included for our approach (QMPSR) to show the dispersion.

It is also worth mentioning that in experiment B, Value-Oriented and QMPSR methods generate a higher number of collided requirements than in experiment A. In experiment B, Value-Oriented and QMPSR methods produce an average of 64.00% (SD=41.88) and 182.83% (SD=33.83) higher collided requirements than in experiment A. However, the experiment B involves a lower number of evaluations for each set of input requirements than in experiment A – i.e., the experiment B presents an average of 40.41% (SD=1.34) less prioritization effort than the experiment A (see the first two rows in Table 9). Thus, these methods are affected by a reduced *NPD* and prioritization effort (number of evaluations for each set of input requirements), especially QMPSR. However, QMPSR generates less collided requirements in all sets of requirements in comparison with all other methods.

With respect to experiment C (see Fig. 7), QMPSR generates less collided requirements than in experiments A and B. For instance, in experiment C, QMPSR generates an average of 5.81% (SD=1.07) collided requirements, while in experiments A and B it obtains 19.9% (SD=3.82) and 55.22% (SD=7.97), respectively. It is also worth noting that in experiment C the average of

prioritization effort is 77.57% (SD=4.85) and 198.05% (SD=5.49) higher than in experiments A and B, respectively (see the first three rows in Table 9). Therefore, it can be stated that when QMPSR is evaluated with a *random PEL*, it generates less collided requirements as the *NPD* and prioritization effort increase. Similarly, we can also see that differences among QMPSR and the rest of methods increase when considering larger sets of requirements.

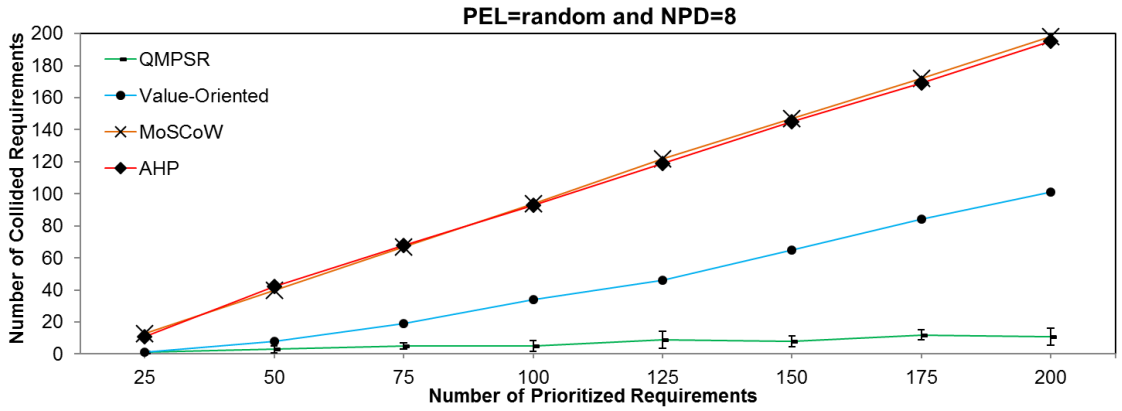

Fig. 7. Experiment C. Number of collided requirements (y axis) for all methods evaluated: QMPSR, Moscow, Value-Oriented and AHP. Values obtained as an average over 10 executions for 25, 50, 75, 100, 125, 150, 175 and 200 prioritized requirements (x axis), considering *PEL=random* and *NPD*=8. Error bars (SD) are included for our approach (QMPSR) to show the dispersion.

As for experiments D, E and F (see Fig. 8, 9 and 10, respectively), they allow to analyze the collisions generated in prioritization methods when there is a *minimum PEL* and an *NPD* of 4, 2 and 8, respectively. Results obtained from these experiments provide empirical evidence that all compared methods generate a similar number of collided requirements for each set of requirements.

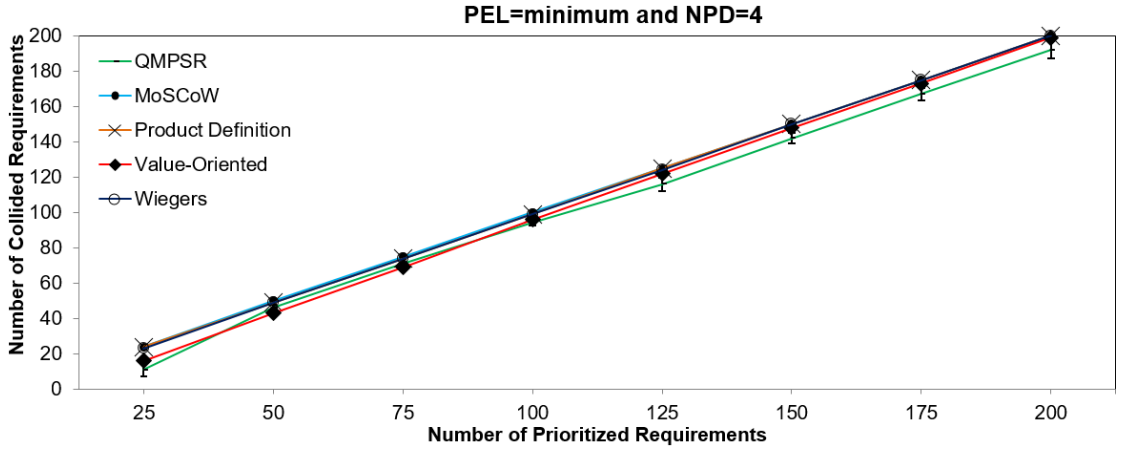

Fig. 8. Experiment D. Number of collided requirements (y axis) for all methods evaluated: QMPSR, Moscow, Value-Oriented, Wiegers and Product Definition. Values obtained as an average over 10 executions for 25, 50, 75, 100, 125, 150, 175 and 200 prioritized requirements (x axis), considering *PEL=minimum* and *NPD*=4. Error bars (SD) are included for our approach (QMPSR) to show the dispersion.

In experiment D (see Fig. 8), QMPSR obtains an average of 87.94% (SD=17.8) collided requirements, while the other methods obtain values reaching 91%. Similarly, in experiment E (see Fig. 9), QMPSR produces an average of 84.59% (SD=5.91) collided requirements,

whereas values obtained from the other methods reach 96%. However, as we can see in Fig. 9, QMPSR generates less collided requirements in all sets of requirements. Finally, in experiment F (see Fig. 10), our method generates an average of 86.92% (SD=16.97) collided requirements, while the other methods get values reaching 96%. In this experiment, QMPSR stands out above the others for the set of 25 requirements, producing 48% collided requirements, while other methods get values reaching 92%.

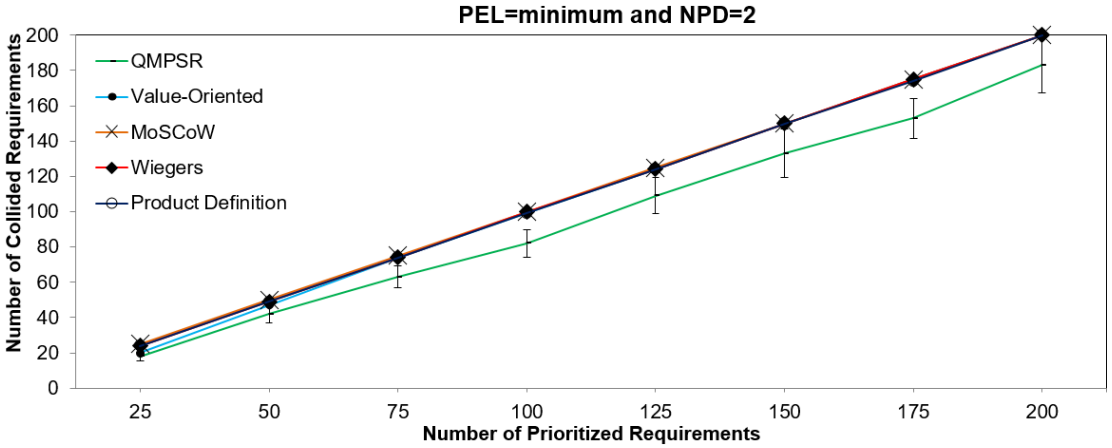

Fig. 9. Experiment E. Number of collided requirements (y axis) for all methods evaluated: QMPSR, Value-Oriented, Moscow, Wiegers and Product Definition. Values obtained as an average over 10 executions for 25, 50, 75, 100, 125, 150, 175 and 200 prioritized requirements (x axis), considering *PEL=minimum* and *NPD*=2. Error bars (SD) are included for our approach (QMPSR) to show the dispersion.

One of the factors that may explain the above-described results is the considerable decrease of prioritization effort in experiments D, E and F. Although experiments A, B and C include the same *NPD* as experiments D, E and F, respectively, experiments D, E and F include only one of the prioritization dimensions defined (randomly identified) to determine the value for each requirement – i.e., they generate less prioritization effort. For example, in experiment D, methods generate an average of 151.98% (SD=4.99) less prioritization effort than in experiment A. Similarly, in experiments E and F, methods generate an average of 50.08% (SD=1.67) and 347.26% (SD=5.54) less prioritization effort than in experiments B and D, respectively. In addition, the configuration in these experiments (using only one of the dimensions to prioritize the requirements) is not representative of a real setting, as methods are not provided with enough information to distinguish the relevance of each requirement, and thus hinder the identification of collided requirements.

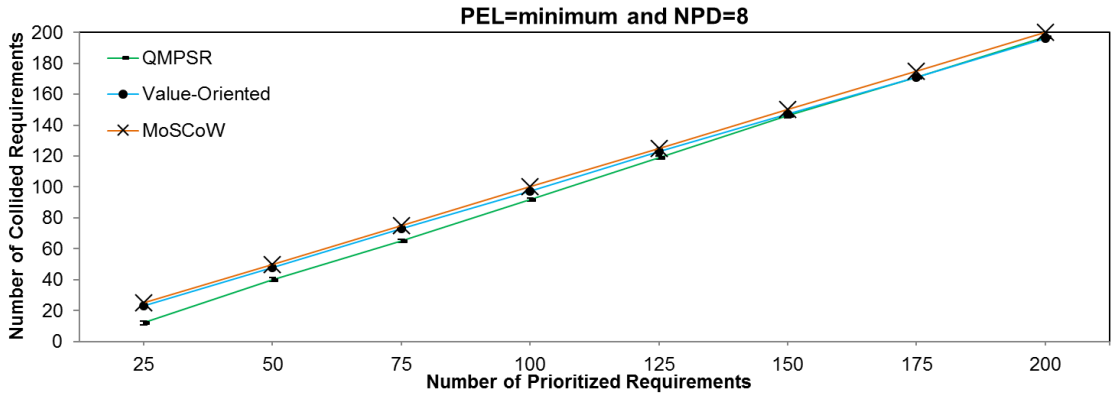

Fig. 10. Experiment F. Number of collided requirements (y axis) for all methods evaluated: QMPSR, Value-Oriented and Moscow. Values obtained as an average over 10 executions for 25, 50, 75, 100, 125, 150, 175 and 200 prioritized requirements (x axis), considering *PEL=minimum* and *NPD*=8. Error bars (SD) are included for our approach (QMPSR) to show the dispersion.

Regarding experiments G, H and I (see Fig. 11, 12 and 13, respectively), they provide the collisions produced by each prioritization method with a *maximum PEL* and an *NPD* of 4, 2 and 8, respectively. Results obtained from these experiments strongly demonstrate that QMPSR outperforms the other compared methods.

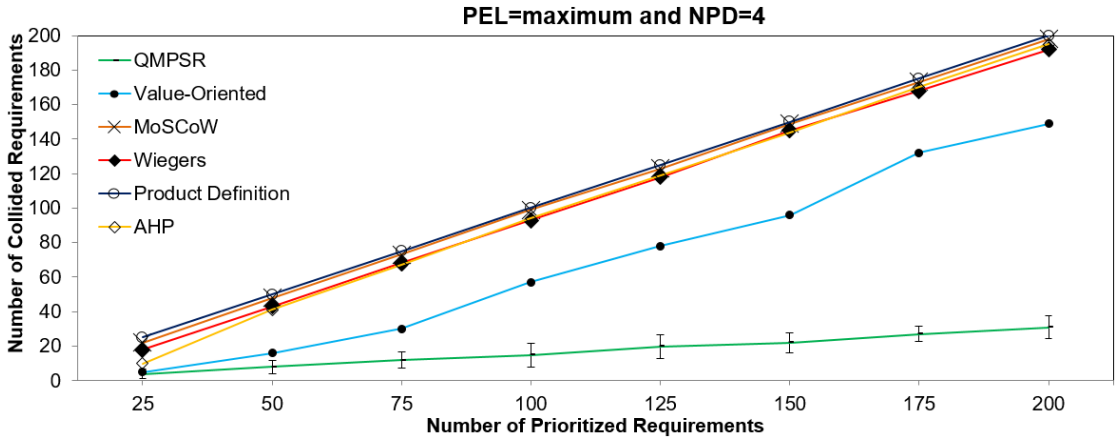

Fig. 11. Experiment G. Number of collided requirements (y axis) for all methods evaluated: QMPSR, Value-Oriented, Moscow, Wiegers, Product Definition and AHP. Values obtained as an average over 10 executions for 25, 50, 75, 100, 125, 150, 175 and 200 prioritized requirements (x axis), considering *PEL=maximum* and *NPD*=4. Error bars (SD) are included for our approach (QMPSR) to show the dispersion.

Results from experiment G (see Fig. 11) show that QMPSR generates less collided requirements for all sets of requirements. Moreover, the QMPSR provides less collided requirements in comparison with other experiments including the same *NPD* (4) but different *PEL* (*random* and *minimum* in experiments A and D, respectively). For example, in experiment G, QMPSR generates an average of 4.33% and 36.63% less collided requirements than in experiments A and D, respectively. However, the experiment G involves a higher number of selected prioritization dimensions to determine the value for each requirement than in experiments A and D – i.e., the experiment G generates an average of 58.79% (SD=3.11) and 300% (SD=0) higher prioritization effort than in experiments A and D, respectively (see the first, fourth and seventh row in Table 9). Therefore, it can be confirmed that when QMPSR is

evaluated with 4 prioritization dimensions considering different *PEL* values, there are less collided requirements as the prioritization effort increases.

Likewise, results from experiment H (see Fig. 12) illustrate that QMPSR, Value-Oriented and AHP methods obtain a similar number of collided requirements for the set of 25 prioritized requirements. Nevertheless, the difference in the number of collided requirements between QMPSR and the other methods becomes more explicit as the number of input requirements increases. In this experiment, QMPSR produces less collided requirements for all sets of requirements, particularly for the sets of 25 and 50, where the average of collided requirements is only 37% (SD=1.41).

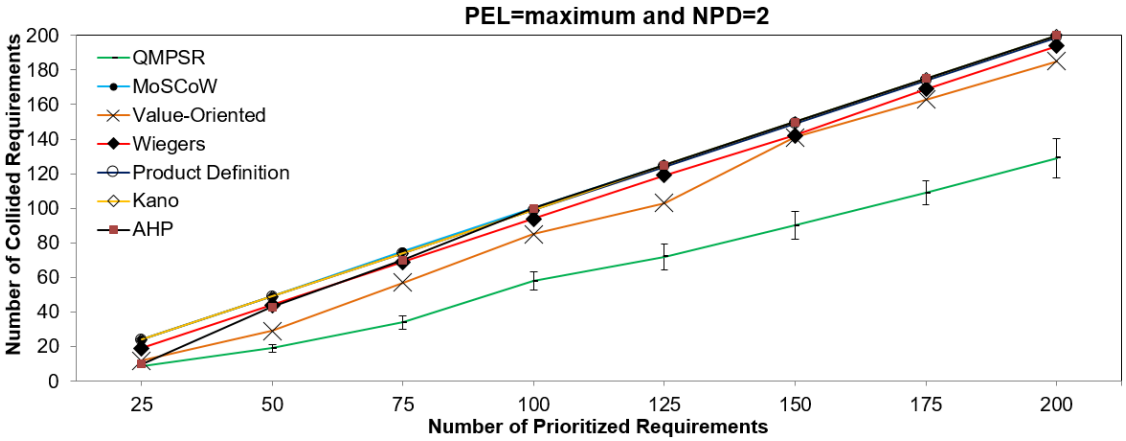

Fig. 12. Experiment H. Number of collided requirements (y axis) for all methods evaluated: QMPSR, Value-Oriented, Moscow, Wiegers, Product Definition, Kano and AHP. Values obtained as an average over 10 executions for 25, 50, 75, 100, 125, 150, 175 and 200 prioritized requirements (x axis), considering *PEL=maximum* and *NPD*=2. Error bars (SD) are included for our approach (QMPSR) to show the dispersion.

Results obtained from the last experiment (see experiment I in Fig. 13) demonstrate that QMPSR also generates less collided requirements for all sets of requirements. In this experiment, our method only produces an average of 5.59% (SD=1.34) collided requirements, while the other methods obtain values reaching 63%. The Value-Oriented method appears as the second best, but for sets of 25, 50 and 75 requirements it shows an average of 20.44% (SD=8.67) collided requirements, whereas QMPSR provides an average of 6% (SD=2) collided requirements for the same set of input requirements.

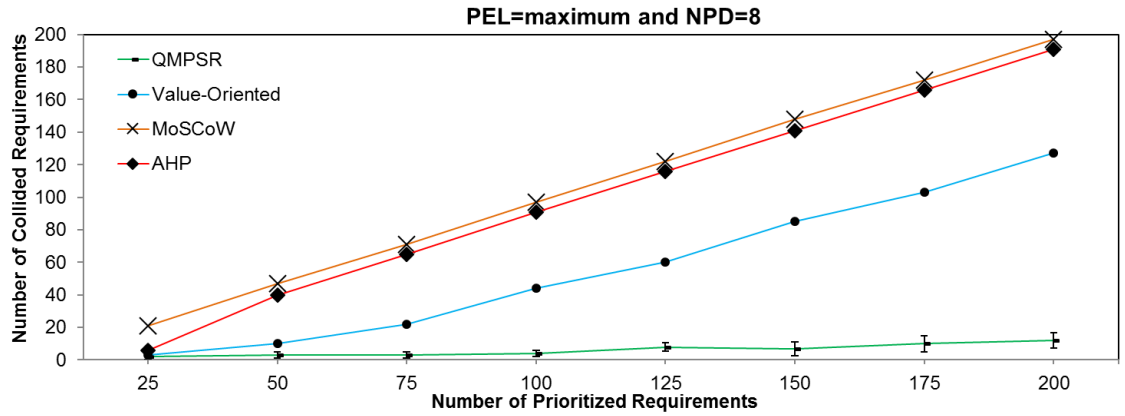

Fig. 13. Experiment I. Number of collided requirements (y axis) for all methods evaluated: QMPSR, Value-Oriented, Moscow and AHP. Values obtained as an average over 10 executions for 25, 50, 75, 100, 125, 150, 175 and 200 prioritized requirements (x axis), considering *PEL=maximum* and *NPD*=8. Error bars (SD) are included for our approach (QMPSR) to show the dispersion.

It can be noted that in experiments G, H and I (*PEL=maximum*), the evaluated methods utilize all prioritization dimensions defined to determine the value for each requirement. Thus, the experiment I produces an average of 200% (SD=0) and 400% (SD=5.49) higher prioritization effort than in experiments G and H, respectively (see the last three rows in Table 9). In general, QMPSR generates less collided requirements in experiment I than in experiments G and H. For example, in experiment I our method produces an average of 191.78% (SD=67.12) and 902.17% (SD=330.05) less collided requirements than in experiments G and H, respectively. Therefore, it can be stated that when QMPSR is arranged with a *maximum PEL*, there are less collided requirements as the *NPD* increases.

We can observe that QMPSR obtains the best results in all experiments involving a *random* or *maximum PEL* and 8 prioritization dimensions. In fact, the average of collided requirements is less than 6% in these cases. Similarly, in experiments with 4 prioritization dimensions and a *random* or *maximum PEL*, the average of collided requirements is less than 20%. By contrast, QMPSR generates a greater number of collided requirements in all experiments with a *minimum PEL*, obtaining an average of around 86% collided requirements, whereas the other methods obtain values reaching 91%.

Table 10

Percentages of Requirements Collisions. Collided requirements generated by the methods through all the experiments.

	Methods						
	QMPSR	**Value-Oriented**	**MoSCoW**	**Wiegers**	**Product Definition**	**AHP**	**Kano**
Mean	46.03%	68.68%	97.03%	93.93%	99.08%	87.42%	98.95%
SD	35.19	24.6	3.68	4.15	0.49	3.45	0
Max	87.94%	96.8%	100%	98.98%	100%	91.24%	98.95%
Min	5.59%	32.24%	88.52%	89.05%	98.64%	82.35%	98.95%

Finally, the results obtained provide enough evidence to answer the previously stated research questions:

- *RQ1*: Results obtained from all experiments demonstrate that QMPSR reduces the number of collided requirements in comparison with all other methods. Our qualitative approach, based on an established relationships among the requirements' aspects and elements, allows to better discriminate between requirements associated with the same priority elements thanks to the Association Factor previously defined, which produces accurate results in the final ranking of requirements.
- *RQ2*: To demonstrate how our approach outperforms the rest of the methods in terms of requirements collisions, Table 10 summarizes the collided requirements produced by the compared methods through all the experiments. In general, maintaining the same prioritization effort for all methods in each experiment, our approach produces less

collided requirements. The results obtained show that QMPSR outperforms all compared methods, obtaining an average of 46.03% (SD=35.19) collided requirements. This best result is followed by Value-Oriented, AHP, Wiegers, MoSCoW, Kano and Product Definition methods, which produce the following averages of collided requirements: 68.68% (SD=24.6), 87.42% (SD=3.45), 93.93% (SD=4.15), 97.03% (SD=3.68), 98.95% (SD=0) and 99.08% (SD=0.49), respectively. It is worth mentioning that the minimum value was obtained by our approach (5.59% of collided requirements) in experiment I where, as previously analyzed, QMPSR achieved the best results under adverse conditions.

- *RQ3:* To demonstrate how our approach overcomes scalability problems as the number of input requirements increases, we analyzed the graphics obtained through all the experiments (Fig. 5-13). We carried out a linear regression for all the methods studied, obtaining satisfactory R squared values in all cases as an indicator of an accurate straight-line fitting (M=0.99, SD=0.01). In this way, the scalability was studied in terms of the slope of the resulting lines for all methods in each experiment, which indicates how each method behaves when the number of input requirement increases. A slope value closer to 1 indicates a linear growth of collisions as the number of input requirement increases. On the other hand, a slop value closer to 0 corresponds to the ideal case to prevent scalability problems, since the slope of the line (first derivative) is almost constant. By contrast, a slope value over 1 indicates a quick growth of collided requirements as the number of input requirements increases (the worst situation). Table 11 reports the slopes of the straight-line fitting for each method through the experiments. As we can see, QMPSR outperforms all compared methods, obtaining an average slope of 0.49 (SD=0.38), which is a slope under 1 and not far away from 0. This best result is followed by Value-Oriented, Wiegers, AHP, Product Definition, MoSCoW and Kano methods, which produce the following averages slopes: 0.78 (SD=0.2), 0.96 (SD=0.02), 0.96 (SD=0.02), 0.99 (SD=0.01), 0.99 (SD=0.01) and 0.99 (SD=0), respectively. It is also worth mentioning the minimum slope value obtained by our approach in experiment I (0.05). In this experiment, the slope of the line becomes almost constant, which is a good indicator of how our method behaves under adverse conditions.

Table 11

Fitting-line Slopes for all Methods. Slopes of the fitting lines for the compared methods in all experiments.

	Methods						
	QMPSR	**Value-Oriented**	**MoSCoW**	**Wiegers**	**Product Definition**	**AHP**	**Kano**
Mean	0.49	0.78	0.99	0.96	0.99	0.96	0.99
SD	0.38	0.2	0.01	0.02	0.01	0.02	0
Max	0.96	0.99	1	0.99	1	0.99	0.99
Min	0.05	0.43	0.97	0.93	0.99	0.93	0.99

7. Threats to Validity

We have presented a set of controlled experiments, where the same data were systematically applied to evaluate all featured algorithms. It is worth mentioning that, according to our experimental framework, we have applied multiple experiments that have been somehow replicated using different conditions before representing and analyzing the results. On the other hand, results were calculated as an average over 10 executions with the same data set. The prioritization effort calculated helps ensure reliable results per experiment.

As for internal validity, the relationship existing among the independent variables and the dependent one can be considered as straightforward according to the numerical interpretation of the results obtained from each experiment. On the one hand, we do not foresee unanticipated events that affected the dependent variable. In general, changes in the dependent variable were due to normal developmental processes operating through the different conditions stated. The only issue to comment is the selection of methods for each experiment, which has been explicitly carried out according to the characteristics of each of them. However, we do not think that this would be a threat for the group design in this case and for the interpretation of the results either. On the other hand, and due to the fact that we are applying the experiments to single groups (prioritization methods), single group threats have to be considered. In this sense, and due to the nature of the data, we do not foresee threats related to history, instrumentation, statistical regression (the subjects were classified into experimental groups based on previous experiments), selection, mortality or ambiguity about direction of causal influence. It is worth mentioning, however, threats related to maturation and testing. As for maturation (the subjects react differently as time passes), we observe different behavior when the number of requirements increases. This is not specifically related to time but to the number of requirements provided as input in different steps of the experiment. Causes and results are explained in Section 6.5, where a discussion about the results obtained is presented. With respect to the testing threat, admittedly the experiments are repeated and the prioritization methods respond differently at different times. However, and due to the nature of the data and the fact that we have not involved humans, this threat is reduced as there is no familiarization with the test nor unintended learning is perceived.

As for external validity, in our approach the number of requirements in sets, used as one of the independent variables, has been increased from a minimal value (25) to a maximal one (200); 25 at a time. However, we think that this does not affect the reliability of the results, as this sample-size range is sufficiently representative to be considered in real settings depending on the project size. In addition, the trend lines included in the experimental charts clearly predict the behavior for broader sample-size sets. Another issue that might affect the external validity is the nature of the data. We have certainly used synthetic data to carry out the experiments. However, this ensures the right processing of the information for all the methods to be compared, due to the inherent heterogeneity of all of them. In general, our approach can be applied to any kind of context regardless of the requirements' domain; the utilization of synthetic data to carry out the evaluations provides evidence of this. Although the generation of synthetic

data in a random way can imply certain variability in the behavior / performance of the methods, different actions were implemented in order to address the sensitivity to input data. In the first place, all the instances susceptible of being generated in a random way were identified, so that the results could not be determined in advance in any case. Secondly, random results were recorded in order to be replicated in all methods. For example, the Prioritization Effort randomly identified for each single requirement was the same for all the methods involved. Finally, the results obtained correspond to an average of 10 executions for each set of requirements. All this helps reduce the variability in the results. Although real requirements in an industrial context might provide more realistic conclusions, this does not represent a definitive limit to generalizability, as the presented method drives the requirements prioritization through the most relevant aspects and elements in a software project, allowing to discern the relevance level of the requirements involved. This is regardless of the context and the domain of the requirements used. In this way, although no real requirements have been used, synthetic data helped obtain comparative findings to observe collided requirements, which is the main objective of the study. This means that most common external validity threats, such as the interaction and treatment of selection, setting and history, do not imply a representative threat in our approach. This is due to the fact that such threats can be reduced by stating and reporting the characteristics of the environment to show the applicability, as we have detailed in previous sections of the paper.

With respect to construct validity, we have based our study on a single dependent variable to measure the collided requirements through different experiments involving different algorithms. The main objective was to demonstrate that QMPSR outperforms other prioritization methods in terms of requirements collisions and scalability when the number of input requirements increases. In this sense, to demonstrate the construct validity we have compared QMPSR with other prioritization methods under the same conditions in each experiment, obtaining results with statistical significance. As for the scalability testing, a linear regression obtaining satisfactory R squared values was used as an indicator to study the scalability in terms of the slope for all methods, indicating how each method behaves when the number of input requirements increases. This guarantees that the experiments and the statistical methods used help corroborate main research questions and obtain the measure pursued to compare our algorithm against the selected methods, which was the objective of the study. In this sense, there is not inadequate preoperational explication of constructs (due to the fact that the construct is no sufficiently defined before it is translated into measures), mono-operation bias (we consider more than one independent variable), confounding constructs and levels of constructs, interaction of different treatments, interaction of testing and treatment or restricted generalizability across constructs. Social threats are not applicable in this case. As for the mono-method bias threat, we use a single type of measure (number of collided requirements). The comparisons among the collided requirements on the different prioritization methods are based on objective and measurable values, reducing the possibility that the experiments may bias the measures. Measurement bias results from poorly measuring the outcome, which is not the case in our proposal.

Finally, conclusion validity can be assumed by considering the statistical evidence presented throughout the paper. As indicated in our reflection on construct validity, the utilization of a single dependent variable drastically reduces specific threats related to the complexity of the construct. In fact, main research questions can be answered considering the results obtained for requirements collisions. Due to the data utilized in our experiments (random synthetic data), results do not fall into threats such as fishing and error rate, nor it is necessary to prove specific issues in this case related to heterogeneity or restriction of range. As for statistical power, we utilized Mann-Whitney-Wilcoxon test to evaluate the difference of the means among the number of collided requirements obtained by QMPSR and the other methods. The power in this test is high enough and the representation through graphical comparisons among the different prioritization methods provides evidence of the differences found. Due to this, we did not use assumptions on the normality of the distribution (threat on violated assumptions of statistical test). The random nature of the data prevents the fishing threat, whereas obtaining objective measures that can be repeated with the same outcome prevents the threat on reliability of measures.

8. Limitations

Our method has 5 main limitations that are described below:

1. The definition and prioritization of an appropriate set of aspects and elements is highly dependent on the expertise of the corresponding decision-makers. This challenges requirements engineering researchers to create theories that explain and predict the acceptance of qualitative elements relevant to drive the requirements prioritization process. Similarly, it is necessary to formalize knowledge about how decision-makers have to think in terms of aspects, elements and requirements. In this way, a method is required to analyze in depth the quality of the prioritization depending on the context, in order to obtain an understanding of the decision-making process. This includes identifying problems that may affect the final classification according to small changes in the input data.

2. The mechanism used to define the priorities of aspects and elements (ordering the aspects according to their priority and assigning a scale of priorities to each element) can be somewhat subjective and arbitrary. In fact, more elaborated and advanced mechanisms could be used to assign these priorities. However, the initial decision was to simplify this process, focusing the discussion on the identification of relationships among the elements of the project's relevant aspects. We aim at promoting formal argumentations regarding the different decisions made at each stage of the process.

3. Another limitation to consider is related to how the priority of aspects and elements affects the final ranking. Notably, changes in the aspects' priorities produce important alterations in the calculation of the final ranking of requirements. This implies that the use of our approach requires a clear definition of priorities from the initial stages of the prioritization process. However, evaluation scales (normalized priority) have been used to minimize this impact in the calculation of the final ranking.

4. It is worth mentioning that our proposal requires the formal establishment of a minimum set of priorities (i.e., business value, preferences of users and customers, organizational strategy, etc.) to conduct the requirements prioritization process. In this way, our method is not appropriate for informal environments that do not require a strict management of prioritization settings. On the other hand, the most appropriate environments to use our proposal correspond to those that require a formal arrangement, or at least a formal agreement in the decisions made by stakeholders, resulting in a better traceability and validation of requirements with respect to their importance in the project.

5. Even though our method does not present limitations regarding the number of aspects and qualitative elements to be managed, the definition of larger sets of aspects and elements could reduce the speed and effectiveness of the prioritization process. To minimize this limitation, it is necessary to consider the cognitive limits related to the number of aspects and elements to be managed during the prioritization process (Riegel & Doerr, 2015).

9. Conclusions and Future Work

This paper presents a qualitative method for prioritizing software requirements, considering aspects and elements that define the relevance of the project's software requirements. It also deals with project priorities in order to verify the ranking accuracy. Our prioritization method, named QMPSR (Qualitative Method for Prioritizing Software Requirements), aims at driving the requirements prioritization process through the use of qualitative elements.

A complete formulation of the method has been presented along the paper, including the most important definitions and properties. Likewise, we have presented a real application scenario using QMPSR, which allows to describe in detail the application of the method and the solving of requirements collisions. In addition, we contribute with an experimental framework to compare our approach with existing prioritization methods, allowing to determine similarities and common patterns among them. The proposed comparative framework achieves a high degree of standardization, homogeneity and reuse for different concepts and components that enable to perform comparisons and evaluations of a same feature in different prioritization methods.

Using our experimental framework, we have defined prioritization effort and collision metrics to accomplish 9 experiments involving different requirements sets, comparing our approach with 6 well-known existing prioritization methods (MoSCoW, Value-Oriented, Wiegers, Product Definition, AHP and Kano). The experiments allowed to carry out an in-depth analysis of collisions generated by the selected prioritization methods.

Results from the experiments provided an answer to the research questions. In this way, it is possible to affirm that, maintaining the same prioritization effort for all methods in each experiment, QMPSR provides better results, proving that QMPSR uniformly outperforms all prioritization methods with a *random* or *maximum PEL* (Prioritization Effort Level), regardless of the *NDP* (Number of Prioritization Dimensions). Likewise, it produces less collided requirements as the *NPD* and input requirements increase. This highlights the capability of QMPSR to adapt to complex and dynamic environments without presenting scalability problems.

Our paper has some implications to theory. Using Gregor's classification (Gregor, 2006), our proposed method can be considered as a Type V theory. That is to say, this paper describes in detail how to carry out the requirements prioritization process in the development of information systems, presenting explicit prescriptions (a novel method) to construct an artifact. Likewise, it is possible to mention that our proposal addresses the conditions that are suggested by (March & Smith, 1995; Hevner, et al., 2004) to produce a contribution of this type of knowledge. More specifically, the utility for a given community of users is described. The mapping of requirements on a scale of importance for the stakeholders is carried out in a novel way, and the results of the various experiments are provided to demonstrate the efficiency.

Our proposal and its results also have implications for professionals and practitioners. Our approach suggests that an organization should strive to formalize the qualitative aspects and elements that drive the prioritization process in each project. This qualitative approach opens the opportunity for organizations to lead and map software projects with their strategic objectives. Qualitative aspects can represent different groups of interest in the organization, which can formalize knowledge and expert judgment to optimize the decision-making process (Macías 2012; Veral & Macías, 2019).

Even in scenarios where our method generates collided requirements, qualitative elements can be used to efficiently support the task of establishing valuation differences among requirements. Likewise, qualitative elements used to prioritize requirements open up the possibility to describe and consider the stakeholders' perspective in agile software development methodologies (Rojas & Macías, 2015). In a similar vein, qualitative elements can also be related to the *sprint*'s objectives in order to help distinguish what requirements will be developed first, or even recalculate the priority of the requirements backlog when new ones are added or their priority changes over time.

As for future work, we expect to extend our approach by proposing a formal language to represent, validate and semantically analyze the description of relevant aspects and elements of a software project. We also expect to integrate more prioritization methods in our framework, evaluating other metrics through new comparative experiments. We want to explore situations in which the priority values of two requirements are very close to each other, in order to extend the analysis to situations in which decision-makers need to understand the real priority (relevance level) of two requirements. Similarly, we will address the issue of collisions in cases when there is dependency between requirements, in order to improve decision-making during the requirements prioritization process. We also expect to carry out real case studies that analyze in depth and qualitatively how difficult it is for decision-makers to think in terms of the main characteristics of our proposal (that is, definition and prioritization of aspects and elements, requirements prioritization driven by the identification of relationships with elements, etc.). Finally, we expect to build an easy-to-use CASE tool (Macías 2008; Macías & Castells 2001; Macías & Castells 2002), which will be used to implement QMPSR in order to automatically provide a semantic support, recommendations and a database of knowledge to carry out comparative experiments in the long-term.

Acknowledgments

This work was partially supported by the Spanish Government [grant number RTI2018-095255-B-I00] and the Madrid Research Council [grant number P2018/TCS-4314]. Likewise, the authors would like to thank Milda Galkute, from the Pontifical Catholic University of Chile, for reviewing the early version of the paper and suggesting useful improvements. Last but not least, authors would like to thank the reviewers for their work, which was essential to improve this paper.

References

Aasem, M., Ramzan, M. & Jaffar, A., 2010. Analysis and optimization of software requirements prioritization techniques. In *2010 International Conference on Information and Emerging Technologies.* Karachi, IEEE, pp. 1-6.

Achimugu, P., Selamat, A. & Ibrahim, R., 2014. A Preference Weights Model for Prioritizing Software Requirements. In *Computational Collective Intelligence. Technologies and Applications.* Springer International Publishing, pp. 30-39.

Achimugu, P., Selamat, A., Ibrahim, R. & Mahrin, M., 2014. A systematic literature review of software requirements prioritization research. *Information and Software Technology,* 56(6), pp. 568-585.

Ahl, V., 2005. An experimental comparison of five prioritization methods: Investigating ease of use, accuracy and scalability.

Alshazly, A. A., Elfatatry, A. M. & Abougabal, M. S., 2014. Detecting defects in software requirements specification. *Alexandria Engineering Journal,* 53(3), pp. 513-527.

Alshehri, S. & Benedicenti, L., 2013. Ranking Approach for the User Story Prioritization Methods. *Journal of Communication and Computer,* Volumen 10, pp. 1465-1474.

Anand, R. & Dinakaran, M., 2017. Multi-voting and binary search tree-based requirements prioritisation for e-service software project development. *Electronic Government, an International Journal,* 13(2), pp. 111-128.

Avesani, P., Bazzanella, C., Perini, A. & Susi, A., 2005. Facing scalability issues in requirements prioritization with machine learning techniques. In *13th IEEE International Conference on Requirements Engineering (RE'05)*, pp. 297-305.

Avesani, P., Ferrari, S. & Susi, A., 2003. Case-based ranking for decision support systems. In *International Conference on Case-Based Reasoning.* Springer Berlin Heidelberg, pp. 35-49.

Azar, J., Smith, R. K. & Cordes, D., 2007. Value-oriented requirements prioritization in a small development organization. *IEEE Software,* Jan.-Feb., 24(1), pp. 32-37.

Babar, M., Ramzan, M. & Ghayyur, S., 2011. Challenges and future trends in software requirements prioritization. In *International conference on computer networks and information technology*, IEEE, pp. 319-324.

Beg, M., Verma, R. & Joshi, A., 2009. Reduction in number of comparisons for requirement prioritization using B-Tree. In *2009 IEEE International Advance Computing Conference,* pp. 340-344.

Berander, P. & Andrews, A., 2005. Requirements prioritization. In *Engineering and managing software requirements.* Springer Berlin Heidelberg, pp. 69-94.

Berander, P., Khan, K. & Lehtola, L., 2006. Towards a research framework on requirements prioritization. *SERPS,* Issue 6, pp. 18-19.

Botta, R. a. B. A., 2007. A prioritization process. *Engineering Management Journal,* 19(4), pp. 20-27.

Cantu, M., Kim, J. & Zhang, X., 2017. Finding hash collisions using MPI on HPC clusters. In *2017 IEEE Long Island Systems, Applications and Technology Conference*, IEEE, pp. 1-6.

Carod, N. & Cechich, A., 2010. Cognitive-driven requirements prioritization: A case study. In *9th IEEE International Conference on Cognitive Informatics (ICCI'10)*, IEEE, pp. 75-82.

Cayola, L. & Macías, J.A., 2018. Systematic Guidance on Usability Methods in User-Centered Software Development. *Information and Software Technology*, 97, pp. 163-175.

Crow, K., 1994. Customer-focused development with QFD*.* In *annual quality congress proceedings-american society for quality control*, pp. 839-839.

Curcio, K., Navarro, T., Malucelli, A. & Reinehr, S., 2018. Requirements engineering: A systematic mapping study in agile software development. *Journal of Systems and Software*, 139, pp. 32-50.

Daneva, M., Damian, D., Marchetto, A. & Pastor, O., 2014. Empirical research methodologies and studies in Requirements Engineering: How far did we come?. *Journal of Systems and Software*, 95, pp. 1-9.

Duan, C., Laurent, P., Cleland-Huang, J. & Kwiatkowski, C., 2009. Towards automated requirements prioritization and triage. *Requirements engineering,* 14(2), pp. 73-89.

Fehlmann, T. M., 2008. New lanchester theory for requirements prioritization. In *2008 Second International Workshop on Software Product Management.* Barcelona, Catalunya, IEEE, pp. 35-40.

Fernandes, J. & Machado, R., 2016. Requirements Negotiation and Prioritisation. In *Requirements in Engineering Projects.* Springer, pp. 119-136.

Firesmith, D., 2004. Prioritizing Requirements. *Journal of Object Technology,* 3(8), pp. 35-48.

Franceschini, F., 2016. *Advanced quality function deployment.* CRC Press.

Fraser, J., 2002. *Setting Priorities.* AdaptivePath Essays.

Gaur, V. & Soni, A., 2010. An integrated approach to prioritize requirements using fuzzy decision making. *International Journal of Engineering and Technology,* 2(4), p. 320.

Greer, D. & Ruhe, G., 2004. Software release planning: an evolutionary and iterative approach. *Information and Software Technology,* 46(4), pp. 243-253.

Gregor, S., 2006. The nature of theory in information systems. *MIS quarterly*, pp. 611-642.

Harker, P., 1987. Incomplete pairwise comparisons in the analytic hierarchy process. *Mathematical Modelling,* 9(11), pp. 837-848.

Hatton, S., 2008. Choosing the right prioritisation method. In *19th Australian Conference on Software Engineering (aswec 2008).* Perth, WA, IEEE, pp. 517-526.

Herrmann, A., & Daneva, M., 2008. Requirements prioritization based on benefit and cost prediction: an agenda for future research. In *2008 16th IEEE International Requirements Engineering Conference,* pp. 125-134. IEEE.

Hevner, A. R., March, S. T., Park, J. & Ram, S., 2004. Design science in information systems research. *Management Information Systems Quarterly,* 28(1), pp. 75-105.

Hopcroft, J. E., 1983. *Data structures and algorithms.* Boston(MA): Addison-Wesley.

Hudaib, A., Masadeh, R., Qasem, M. & Alzaqebah, A., 2018. Requirements Prioritization Techniques Comparison. *Modern Applied Science,* 12(2), p. 62.

Ibrahim, O. & Nosseir, A., 2016. A Combined AHP and Source of Power Schemes for Prioritising Requirements Applied on a Human Resources. *In MATEC Web of Conferences,* Volumen 76, p. 04016.

Ishizaka, A. & Labib, A., 2009. Analytic hierarchy process and expert choice: Benefits and limitations. *OR Insight,* 22(4), pp. 201-220.

Kaiya, H., Horai, H. & Saeki, M., 2002. AGORA: Attributed goal-oriented requirements analysis method. In *Proceedings IEEE joint international conference on requirements engineering*, IEEE, pp. 13-22.

Kano, N., Nobuhiku, S., Fumio, T. & Shinichi, T., 1984. Attractive quality and must-be quality. *Journal of the Japanese Society for Quality Control (in Japanese),* April, 14(2), p. 39–48.

Karlsson, J., 1996. Software requirements prioritizing. *Proceedings of the Second International Conference,* April.pp. 110-116.

Karlsson, J. & Ryan, K., 1997. A cost-value approach for prioritizing requirements. *in IEEE Software,* Sep/Oct, 14(5), pp. 67-74.

Karlsson, J., Wohlin, C. & Regnell, B., 1998. An evaluation of methods for prioritizing software requirements. *Information and Software Technology,* 39(14), pp. 939-947.

Kim, J., Geum, Y. & Park, Y., 2017. Integrating customers' disparate technology readiness into technological requirement analysis: an extended Kano approach. *Total Quality Management & Business Excellence,* 28(5-6), pp. 678-694.

Kwon, J. J., Hong, J. E. & Chung, L., 2016. Collision detection and resolution of hazard prevention actions in safety critical systems. *Journal of Systems and Software,* Volumen 118, pp. 1-18.

Lauesen, S., 2002. *Software Requirements: Styles & Techniques*. Addison-Wesley Professional, Harlow.

Lima, D., Freitas, F., Campos, G. & Souza, J., 2011. A fuzzy approach to requirements prioritization. In *International Symposium on Search Based Software Engineering.* Szeged, Hungary, Springer Berlin Heidelberg, pp. 64-69.

Liu, X., Sun, Y., Veera, C. S., Kyoya, Y., & Noguchi, K., 2006. Priority assessment of software process requirements from multiple perspectives. *Journal of Systems and Software,* 79(11), pp. 1649-1660.

Logue, K. & McDaid, K., 2008. Agile release planning: Dealing with uncertainty in development time and business value. In *15th Annual IEEE International Conference and Workshop on the Engineering of Computer Based Systems (ecbs 2008).* Belfast, IEEE, pp. 437-442.

Ma, Q., 2009. The effectiveness of requirements prioritization techniques for a medium to large number of requirements: a systematic literature review (Doctoral dissertation, Auckland University of Technology).

Macías, J.A., 2008. Intelligent Assistance in Authoring Dynamically-Generated Web Interfaces. *World Wide Web – Internet and Web Information Systems*, 11(2), pp. 253-286.

Macías, J.A., 2012. Enhancing Interaction Design on the Semantic Web: A Case Study. *IEEE Transactions on Systems Man and Cybernetics Part C – Applications and Reviews*, 42(6), pp. 1365-1373.

Macías, J.A. & Castells, P., 2002 Tailoring Dynamic Ontology-Driven Web Documents by Demonstration. In *Proceedings of Sixth International Conference on Information Visualisation* (IV'2002), IEEE, pp. 535-540.

Macías, J.A. & Castells, P., 2001. A Generic Presentation Modeling System for Adaptive Web-based Instructional Applications. In *Proceedings of ACM Conference on Human Factors in Computing Systems* (CHI'2001), ACM, pp. 349-350.

March, S. T. & Smith, G. F., 1995. Design and natural science research on information technology. *Decision support systems*, 15(4), pp. 251-266.Moran, A., 2015. Strategy, Implementation, Organisation and People. *Managing Agile.* Springer International Publishing, p. 266.

Pergher, M., & Rossi, B., 2013. Requirements prioritization in software engineering: A systematic mapping study. *In 2013 3rd International Workshop on Empirical Requirements Engineering* (EmpiRE), pp. 40-44. IEEE.

Perini, A., Susi, A. & Avesani, P., 2013. A machine learning approach to software requirements prioritization. In *IEEE Transactions on Software Engineering,* April, 39(4), pp. 445-461.

Pitangueira, A. M., Maciel, R. S. P., de Oliveira Barros, M., & Andrade, A. S., 2013. A systematic review of software requirements selection and prioritization using SBSE approaches. In *International Symposium on Search Based Software Engineering*, pp. 188-208. Springer, Berlin, Heidelberg.

Pitangueira, A., Maciel, R. & Barros, M., 2015. Software requirements selection and prioritization using SBSE approaches: A systematic review and mapping of the literature. *Journal of Systems and Software,* Volumen 103, pp. 267-280.

Pohl, K., 2010. *Requirements engineering: fundamentals, principles, and techniques*. Springer Publishing Company, Incorporated.

Pohl, K. & Rupp, C., 2011. *Requirements engineering fundamentals: A study guide for the certified professional for requirements engineering exam – Foundation level – IREB compliant.* 1st ed. Rocky Nook.

Racheva, Z., Daneva, M. & Buglione, L., 2008. Supporting the dynamic reprioritization of requirements in agile development of software products. In *2008 Second International Workshop on Software Product Management.* Barcelona, Catalunya, IEEE, pp. 49-58.

Racheva, Z., Daneva, M., Herrmann, A. & Wieringa, R., 2010. A conceptual model and process for client-driven agile requirements prioritization. In *2010 Fourth International Conference on Research Challenges in Information Science (RCIS).* Nice, France, IEEE, pp. 287-298.

Riegel, N. & Doerr, J., 2015. A systematic literature review of requirements prioritization criteria. In *Requirements Engineering: Foundation for Software Quality.* Springer International Publishing, pp. 300-317.

Rojas, L. A. & Macías, J. A., 2015. An Agile Information-Architecture-Driven Approach for the Development of User-Centered Interactive Software. In *Proceedings of the XVI International Conference on Human Computer Interaction.* New York, NY, USA, ACM, p. 8.

Saaty, T., 1980. *The Analytic Hierarchy Process.* New York: McGraw-Hili.

Saaty, T. L., 2008. Relative measurement and its generalization in decision making why pairwise comparisons are central in mathematics for the measurement of intangible factors the analytic hierarchy/network process. *Revista de la Real Academia de Ciencias Exactas, Fisicas y Naturales,* 102(2), pp. 251-318.

Sánchez, E. & Macías, J.A., 2019. A Set of Prescribed Activities for Enhancing Requirements Engineering in the Development of Usable e-Government Applications. *Requirements Engineering*, 24(2), pp. 181-203.

Schön, E., Thomaschewski, J. & Escalona, M., 2017. Agile requirements engineering: a systematic literature review. *Computer Standards & Interfaces,* Volumen 49, pp. 79-91.

Shao, F., Peng, R., Lai, H. & Wang, B., 2017. DRank: A semi-automated requirements prioritization method based on preferences and dependencies. *Journal of Systems and Software,* Volumen 126, pp. 141-156.

Stapleton, J., 1997. *DSDM, dynamic systems development method: the method in practice.* Cambridge University Press.

Tonella, P., Susi, A. & Palma, F., 2013. Interactive requirements prioritization using a genetic algorithm. *Information and software technology,* 55(1), pp. 173-187.

Veral, R. & Macías, J.A., 2019. Supporting User-Perceived Usability Benchmarking Through a Developed Quantitative Metric. *International Journal of Human-Computer Studies*, 122, pp. 184-195.

Vilela, J., Castro, J., Martins, L. & Gorschek, T., 2017. Integration between requirements engineering and safety analysis: A systematic literature review. *Journal of Systems and Software*, 125, pp. 68-92.

Wiegers, K., 1999. First things first: prioritizing requirements. *Software Development,* 7(9), pp. 48-53.

Wiegers, K. & Beatty, J., 2013. *Software requirements.* Pearson Education.

Wieringa, R., 2009. Design science as nested problem solving. In *Proceedings of the 4th international conference on design science research in information systems and technology* (p. 8). ACM.

Wieringa, R. J., 2014. *Design science methodology for information systems and software engineering*. Springer.

Wohlin, C., Runeson, P., Höst, M., Ohlsson, M. C., Regnell, B., & Wesslén, A., 2012. *Experimentation in software engineering.* Springer Science & Business Media.

Zalazar, A. S., Ballejos, L. & Rodriguez, S., 2017. Analyzing Requirements Engineering for Cloud Computing. In *Requirements Engineering for Service and Cloud Computing.* Springer International Publishing, pp. 45-64.

Zhang, H., Li, J., Zhu, L., Jeffery, R., Liu, Y., Wang, Q., & Li, M., 2014. Investigating dependencies in software requirements for change propagation analysis. *Information and Software Technology,* 56(1), pp. 40-53.

Zhou, Q. & Liao, X., 2012. Collision-based flexible image encryption algorithm. *Journal of Systems and Software,* 85(2), pp. 400-407.