A Risk-Based Approach for Selecting Software Requirements

Aruan Amaral
Informatics Center (CI)
Federal University of Paraíba
João Pessoa - PB - Brazil
aruangalves@ppgi.ci.ufpb.br

Gledson Elias
Informatics Center (CI)
Federal University of Paraíba
João Pessoa - PB - Brazil
gledson@ci.ufpb.br

ABSTRACT

In incremental development approaches, there is a great interest in delivering system releases on-time and on-budget, raising the satisfaction level of the stakeholders involved in the development process. Thus, the software requirements selection process has a key role in identifying a good-enough or even an optimal subset of candidate requirements, which can balance trade-offs among critical aspects, such as project budget, requirements costs, customers' preferences and their importance. Despite relevant contributions, current proposals do not address software risks involved in the development process, which represents another key aspect that can deeply impact on project cost and stakeholders' satisfaction. In such a direction, this paper proposes a risk-based approach for selecting software requirements, in which a risk analysis is incorporated for estimating the impact of risks in the cost of the next release requirements and stakeholders' satisfaction. Evaluation results based on a pilot use case reveal the potential practical applicability of the proposed approach.

CCS Concepts

• **Software and its engineering→Requirements analysis** • *Software and its engineering→Risk management*.

Keywords

Next release problem; software requirements; risk management.

1. INTRODUCTION

In the field of Software Engineering, it is not a recent idea the adoption of incremental development processes based on iterative system deliveries, in which each new release is enhanced with some additional functionalities, mapped from a subset of candidate requirements that hold some interest to the stakeholders involved in the development process. Even in the 1970s and 1980s, decades mostly dominated by waterfall development, great enthusiasts of incremental approaches already existed. Indeed, although many view iterative and incremental development as a modern practice, its practiced and published roots go back decades [1]. However, only more recently, enabled by the Internet, group facilitation and distant coordination within open source software communities, iterative and incremental processes have become mainstream in software industry, providing a view of software development and evolution that is incremental, iterative, ongoing, interactive, and sensitive to social and organizational circumstances [2].

In such iterative and incremental processes, during the requirements elicitation process, the needs and interests of every stakeholder should be mapped into a set of candidate requirements to be developed on the next iteration or in a later system release. However, it is widely known that during a software process iteration, schedule and budget limits exist and, ideally, such limits should not be exceeded [2]. Thus, considering a scenario in which the set of candidate requirements surpasses the budget available for the next system release, the development team must face the problem of deciding on which requirements should be prioritized and selected to be delivered on the next release. Besides the necessary effort on negotiating such requirements with the stakeholders, the development team also find difficulty on selecting an appropriate subset of candidate requirements, which can balance trade-offs among critical aspects, such as project budget, requirements costs, customers' preferences and their importance, keeping project costs under control and raising the satisfaction level of the stakeholders involved in the software product.

In such a scenario, without the support of systematic decision-making approaches, the software requirements selection process becomes a complex, challenging and error-prone task. Therefore, the adoption of manual, ad-hoc approaches are impractical for selecting requirements that maximize stakeholder satisfaction and minimize project costs. Such impracticability can be perceived by the large amount of data and conflicts of interest among stakeholders, imposing to the development team the responsibility and the brutal effort to conciliate and balance the trade-offs during the decision-making process, which turns out to be even harder when the set of candidate requirements becomes larger and larger.

In order to alleviate the decision-making effort, information related to candidate requirements must be measured and quantified for making associated data computable. A notable approach on this direction is the cost-importance model [3]. This model aims to maximize software quality, perceived through the satisfaction level of the stakeholders. Besides, it also attempts to minimize costs and delivery time as much as possible. In such an approach, a systematic requirements review process is conducted together with stakeholders, collecting data and generating a two-dimensional diagram that address requirements costs and importance from the point of view of the stakeholders.

Later, in the early 2000s, the software requirements selection process was defined and modeled as the Next Release Problem (NRP) [4]. In the literature, many different proposals that characterize and solve this problem from different and complimentary points of view can be found [5] [6] [7] [8] [9]. Initially, the main advancements are represented by proposals that

SBSI 2017, June 5–8, 2017, Lavras, Minas Gerais, Brazil.

focus on finding the better solution that keeps project budget under control and raises the stakeholders' satisfaction. Later, the concept of a pre-allocated budget limit was discarded in favor of offering not only a single solution but a set of good candidate solutions, independent of budget limits, making more flexible and customizable the decision-making process [5].

Despite relevant contributions, it can be noted that most of current proposals mainly focus on the traditional model of evaluating project budget, requirements costs, customers' preferences and their importance, ignoring other important aspects, such as software risks, including those associated to software project, software product and organization, which can deeply impact positively or negatively on the requirements costs and stakeholders' satisfaction for the next release.

Consequently, the incorporation of a risk analysis in the modeling of the next release problem seems to be an important contribution and evolution. Such rationale can be reinforced by the fact that, on the one hand, the adoption of a risk management process is a significant reason related to software project success [10], and, on the other hand, its absence is a significant reason associated to software project failures [11], once it obstructs project managers and their development teams from assessing the specific points of failure in their respective software projects. For instance, a study done on 50,000 completed software projects reveals that 53% of these projects had some troubles on delivery, including an average budget overrun of 56% [12]. Therefore, the requirements costs must still be kept on the modeling of a next release planning. As a consequence, requirements costs and risk management must be considered simultaneously.

In such a direction, this paper proposes a risk-based approach for selecting software requirements for the next software release, in which a risk analysis is incorporated for estimating the impact of risks in the costs of the selected requirements and stakeholders' satisfaction. In the proposed approach, candidate requirements are associated to identified software risks, which in turn are related to risk mitigation techniques. Then, based on the probability and severity of the identified risks, together with the cost of applying each mitigation technique, the proposed approach estimates the impact of the risks on both requirements costs and stakeholders' satisfaction. Also, as a mean to conciliate and balance conflicts of interest among stakeholders, as adopted in already existing proposals, the proposed approach also considers customers' preferences and their importance to the development organization.

The remainder of this paper is organized as follows. Section 2 presents the proposed approach in details. Section 3 discusses the main related work, evincing the contribution of the proposed approach. In Section 4, a pilot use case is presented in order to evaluate the proposed approach and evaluation results reveal its potential practical applicability. In conclusion, Section 5 discusses some final considerations and future work.

2. RISK-BASED APPROACH

Initially, the scenario of the next release problem must be characterized, making possible to be handled as a computable model and to obtain quantified solutions that represent their quality. In this sense, Figure 1 presents the main steps involved in the proposed risk-based approach and, besides, illustrates the information adopted as input and produced as output in each step.

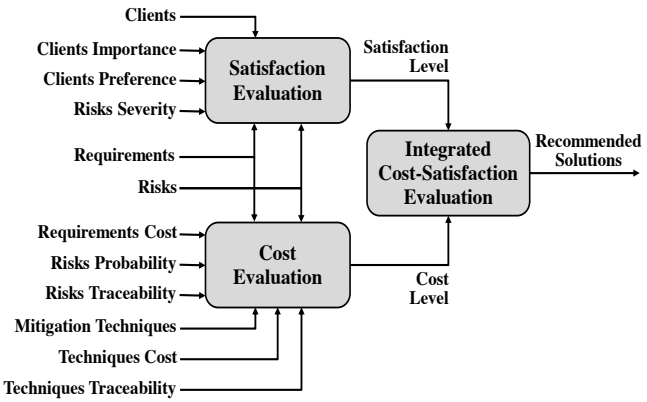

Figure 1. Steps of the proposed approach.

In the proposed approach, the **clients** are represented by the set *U*, as indicated in Equation 1. On the next release, these clients have interest on the set of candidate **requirements**, represented by the set *RQ* and defined in Equation 2.

$$U = \{u_1, u_2, \ldots, u_p\} \quad (1)$$

$$RQ = \{rq_1, rq_2, \ldots, rq_n\} \quad (2)$$

It is also known that just a subset of the candidate requirements *RQ* will be selected for the next release. As such, in order to indicate the selection of each requirement rq_i, each possible **recommended solution** is represented by the set *X*, as defined in Equation 3. Each term x_i assumes values 1 or 0 to denote that the corresponding requirement rq_i has been selected or not, respectively.

$$X = \left\{ x_i \,\middle|\, \exists_{rq_i}\, rq_i \in RQ \ \wedge\ x_i \in \{0,1\} \right\} \quad (3)$$

Clients must define a set of requirements preferences, called **clients preference**, assigning higher preferences to requirements of most importance. Thus, for each client u_l, it defines a degree of preference attributed for each requirement rq_i. As indicated in Equation 4, such a preference is represented by the relationship $S_{U,RQ}$, composed by terms called $s_{l,i}$, which assumes values in the interval $[0,1] \in \mathbb{R}$. Note that the value one indicates that client u_l has the maximum interest on requirement rq_i, while the value zero denotes the complete lack of interest.

$$S_{U,RQ} = \left\{ s_{l,i} \,\middle|\, \exists_{u_l}\, u_l \in U \ \wedge\ \exists_{rq_i}\, rq_i \in RQ \ \wedge\ s_{l,i} \in [0,1] \right\} \quad (4)$$

Consequently, clients may have distinct or even conflicting interests about the requirements to be selected in the next release. To deal with such conflicts of interest, the organization responsible by developing the software product must adopt a conciliating strategy for balancing conflicting preferences. In the proposed approach, such a strategy is based on the concept of **clients importance** from the point of view of the development organization, indicating how important each client u_l is to its business strategy. The clients importance is represented by the set *E*, as shown in Equation 5, in which each term e_l can assume values in the interval $(0,1] \in \mathbb{R}$, indicating the importance associated to the respective client u_l. Differently from requirements preferences, clients importance cannot be completely discarded and so must be different from zero.

$$E = \left\{ e_l \,\middle|\, \exists_{u_l}\, u_l \in U \ \wedge\ e_l \in (0,1] \right\} \quad (5)$$

In order to evaluate the next release cost, the **requirements cost** is a set of values associated to each requirement rq_i, which is

represented by the set *RQC* in Equation 6. It is important to highlight that the development team, based on some kind of cost estimation method adopted during project planning, supplies the requirements cost.

$$RQC = \{rqc_i \mid \exists_{rq_i} rq_i \in RQ \ \wedge \ rqc_i > 0\} \quad (6)$$

The information regarding requirements costs is not sufficient for decision making about the best selection of requirements for the next version of the system. There are other important factors that define the degree of success of the subset of requirements delivered on the next release [13]. Besides the cost estimation, other risk factors associated to the requirements engineering process can arise during the planning of the next system release. These factors can represent the failure or success of the new release according to the final users' or clients' view [13] [14].

In such a direction, the proposed approach deals with risks associated with software requirements. As an example of software risk, it can be indicated requirements changes as inevitable in all large software projects due to constant evolution in the clients' business interest. Besides, as an additional example, it can be signalized the lack of comprehension on the requirements definition, including a misunderstanding of the stakeholders' real interests or even an omission of important features or constraints associated to requirements. This lack of comprehension or incompleteness can be related to a specific context, where the chosen language to define the requirements has a special meaning in the client's business domain. Besides that, the software product can experience an uncontrolled growth due to problems during the definition of its constraints and scope. This can be a result of vague or abstract requirements descriptions, sometimes deliberately produced in order to find a common ground among stakeholders with conflicting views [15] [16] [17].

The requirements elicitation should not be very ambitious, but must be done in a realistic and achievable way. Also, a lack of traceability during the requirements validation stage can also become a risk factor. From the final users' point of view, they might also reject the software product when they are not appropriately involved during the software development process, generating a resistance to changes. Indeed, several other more specific risk factors can be considered. Such risks generate a series of possible undesirable events that can impact the satisfaction level perceived by clients and the final cost of the delivered next release, all of them related to an inadequate selection of requirements [18].

Consequently, taking into account the changeable context related to software development processes, numerous features and constraints related to software project, software product, development environment, deployment hardware and software platforms, as well as development team skills and availability, and finally managerial and organizational aspects [17], have the potential to direct or indirectly affect the candidate software requirements [14].

Usually, a risk is defined as a material or financial loss, or any other event that must be avoided in a risk management process [13] [19] [20]. Every risk is associated with a *severity* value, which indicates the consequences of the risk event and its associated loss or impact [19] [20]. Besides, every risk also has a *probability* value, which indicates how likely this undesirable event could happen.

In the proposed approach, the **risks** are defined by the set *RK*, as indicated in Equation 7. As mentioned, each risk rk_j has associated probability and severity values. However, given the difficulty of associating accurate values to probability and severity, in the proposed approach, a fuzzy-driven representation is adopted, in which textual terms are employed to classify probability and severity as discrete values represented in Table 1.

$$RK = \{rk_1, rk_2, \dots, rk_m\} \quad (7)$$

Table 1. Risk probability and severity values.

Class	Very Low	Low	Medium	High	Very high
Value	0.05	0.25	0.50	0.75	0.95

As can be noticed in Table 1, a five-point value Likert scale has been adopted because it has been a common measure used on the literature [21]. Thus, the values assigned to risk probability and severity can only assume values in the set *FT*, defined in Equation 8.

$$FT = \{0.05, 0.25, 0.50, 0.75, 0.95\} \quad (8)$$

In the proposed approach, the **risks probability** and **risks severity** are defined respectively by the sets *RKP* and *RKS*, as represented in Equations 9 and 10. Note that the corresponding terms rkp_j and rks_j can only assume values represented by the fuzzy-terms in Table 1 and formalized by the set *FT* in Equation 8.

$$RKP = \left\{rkp_j \mid \exists_{rk_j} rk_j \in RK \ \wedge \ rkp_j \in FT\right\} \quad (9)$$

$$RKS = \left\{rks_j \mid \exists_{rk_j} rk_j \in RK \ \wedge \ rks_j \in FT\right\} \quad (10)$$

As a mean to represent the traceability among requirements and software risks, called in the proposed approach as **risks traceability**, each requirement rq_i might be associated with one or more software risks rk_j. Therefore, as indicated in Equation 11, the relationship $RT_{RQ,RK}$ is adopted to represent such a traceability, in which the term $rt_{i,j}$ denotes the traceability between the requirement rq_i and the risk rk_j, assuming a value equal to one or zero to indicate its existence or not.

$$RT_{RQ,RK} = \left\{rt_{i,j} \mid \exists_{rq_i} rq_i \in RQ \ \wedge \ \exists_{rk_j} rk_j \in RK \ \wedge \ rt_{i,j} \in \{0,1\}\right\} \quad (11)$$

In a risk management process, in the case of a risk occurrence, it ought to be adopted **mitigation techniques** [20] to reduce or eliminate the consequences of the risk. In the proposed approach, mitigation techniques are defined by the set *T*, defined in Equation 12.

$$T = \{t_1, t_2, \dots, t_q\} \quad (12)$$

Mitigation techniques have associated costs related to their adoption as defined in Equation 13 by the set *TC*, called in the proposed model as **techniques cost**. It is important to emphasize that, based on some kind of cost estimation method adopted during project planning, the development team ought to provide the cost value tc_k associated to each mitigation technique t_k.

$$TC = \{tc_k \mid \exists_{t_k} t_k \in T \ \wedge \ tc_k > 0\} \quad (13)$$

During the development process, each mitigation technique can help in mitigating one or more risks. As a mean to represent the traceability among risks and mitigation techniques, as formalized in Equation 14, it is also necessary to define the relationship $TT_{RK,T}$, called **techniques traceability**, in which each term $tt_{j,k}$ associates the risk rk_j to the mitigation technique t_k, assuming values equal to one or zero to indicate the presence or absence of the association, respectively.

$$TT_{RK,T} = \left\{ tt_{j,k} \;\middle|\; \exists_{rk_j}\, rk_j \in RK \;\wedge\; \exists_{t_k}\, t_k \in T \;\wedge\; tt_{j,k} \in \{0,1\} \right\} \quad (14)$$

Generally, in risk management processes, mitigation techniques are classified in two different types:

- **Preventive:** attempts to avoid the risk occurrence. It is applied during the whole release implementation, regardless of the risk event happens or not.
- **Corrective:** attempts to mitigate or even eliminate the risk consequence. It is applied after occurring the risk event, during a state of emergency, allowing the software project to return to the expected state, and so eliminating or reducing the consequences associated to one or more risks.

2.1 Satisfaction evaluation

After the definition of every concept related to the risk-based next release problem, the **satisfaction level** can be defined in Equation 15. It indicates the satisfaction perceived by clients, taking into account the subset of selected requirements. Note that the satisfaction level is modeled by the total sum for each client u_l and requirement rq_i, considering the product among the following terms: *(i)* the preference level $s_{l,i}$ that client u_l has in relation to requirement rq_i; *(ii)* the importance level e_l that the development organization assigned to the client u_l; *(iii)* the severity-based software risks measure imp_i associated to requirement rq_i, which is discussed in the next paragraph; and *(iv)* the selector x_i that represents the selection or not of the requirement rq_i in the evaluated solution.

$$S = \sum_{u_l \in U} \sum_{rq_i \in RQ} s_{l,i} \cdot e_l \cdot imp_i \cdot x_i \quad (15)$$

Taking into account the severity-based software risks measure associated to requirement rq_i, represented in Equation 15 by the term imp_i, its value is calculated based on Equation 16, which establishes the relation among the following terms: *(i)* the traceability $rt_{i,j}$ among the requirement rq_i and its associated software risk rk_j; and *(ii)* the risk severity (rks_j) associated to risk rk_j.

$$imp_i = \sum_{rk_j \in RK} rks_j \cdot rt_{i,j} \quad (16)$$

Note that the proposed approach adopts the premise that software processes must first deal with the most critical risks as a mean to maximize the chances of the software project be successful, leading to better satisfaction levels. In such a direction, the proposed approach favors the selection of requirements associated to more severe risks, and, inversely, penalizes the selection of requirements associated to less severe risks. Indeed, usually, risk management processes focus on critical risks, mitigating or avoiding the occurrence of undesirable events that might compromise the clients' satisfaction [10].

2.2 Cost evaluation

In order to represent the **cost level**, Equation 17 is defined by the total sum among the development cost (rqc_i) together with the additional risk management cost (rkc_i) for each requirement rq_i. As defined, the risk management cost represents a penalty in the total cost of the next release, which is an usual strategy in real software projects dealing with risks as an additional cost [21]. As can be noticed, development and risk management costs associated to a given requirement rq_i are only considered if the requirement is selected for the next release, which is represented in Equation 17 by the term x_i.

$$C = \sum_{rq_i \in RQ} (rqc_i + rkc_i) \cdot x_i \quad (17)$$

The risk management cost rkc_i for a given requirement rq_i can be defined by Equation 18. As can be noticed, it is modeled by the total sum of the risk management cost $rkc_{i,j}$ for each particular risk rk_j associated to requirement rq_i.

$$rkc_i = \sum_{rk_j \in RK} rkc_{i,j} \quad (18)$$

It is important to remember that a given risk mitigation technique t_k might be associated to one or more risks rk_j. Thus, it is reasonable to adopt the premise that the cost tc_k of applying a given technique t_k can be divided among all risks rk_j that make use of the technique. Besides, it is also defined that each requirement rq_i can be associated to one or more risks rk_j, which in turn might be associated to one or more mitigation techniques t_k. This relationship information is broken down into a new term called $tc_{i,j,k}$. Now, as illustrated in Equation 19, the risk management cost $rkc_{i,j}$ for each particular risk rk_j associated to requirement rq_i is defined by the ratio between the following terms: *(i)* the total sum of the costs $tc_{i,j,k}$ of applying each technique t_k related to the given risk rk_j and requirement rq_i, introduced later in Equation 20; and *(ii)* the number of times ta_k that technique t_k is applied in all requirements rq_i and risks rk_j, discussed later in Equation 22.

$$rkc_{i,j} = \sum_{t_k \in T} \frac{tc_{i,j,k}}{ta_k} \quad (19)$$

As cited, the term $tc_{i,j,k}$ represents the cost of applying a given technique t_k related to a given risk rk_j and requirement rq_i. In Equation 20, if exists the relationship among requirement and risk ($rt_{i,j}$) and also among risk and technique ($tt_{j,k}$), the term $tc_{i,j,k}$ is defined based on the cost of the technique tc_k and the probability $tp_{j,k}$ of applying the technique t_k to the risk rk_j.

$$tc_{i,j,k} = tp_{j,k} \cdot rt_{i,j} \cdot tt_{j,k} \cdot tc_k \quad (20)$$

In turn, the term $tp_{j,k}$ is given by Equation 21, differentiating technique types. The cost for preventive techniques is integrally considered as they are applied regardless of risk events happen or not. In contrast, the cost for corrective techniques depends on risk probabilities (rkp_j) as they are not applied until the occurrence of risk events, which are uncertain during development processes. Thus, the cost of corrective techniques is defined as an estimation. Risk probability and severity are separately considered as risk mitigation techniques are also considered herein, meaning that risk events do not necessarily cause expected consequences, which are reduced or eliminated with the use of mitigation techniques [22].

$$tp_{j,k} = \begin{cases} 1 & if\ \textbf{\textit{preventive}}\ technique, \\ rkp_j & if\ \textbf{\textit{corrective}}\ technique \end{cases} \quad (21)$$

Now, returning to term ta_k, defined in Equation 22, it denotes the number of times that technique t_k is applied in all combinations among requirements and risks ($rt_{i,j}$) and also among such risks the technique in question ($tt_{j,k}$).

$$ta_k = \sum_{rq_i \in RQ} \sum_{rk_j \in RK} x_i \cdot rt_{i,j} \cdot tt_{j,k} \quad (22)$$

At this point, the risk management cost rkc_i is completely defined spreading the cost of each technique t_k among every risk that makes use of the technique. To make clear this strategy, consider the following example, based on the traceability between requirements, risks and mitigation technique illustrated in Figure 2.

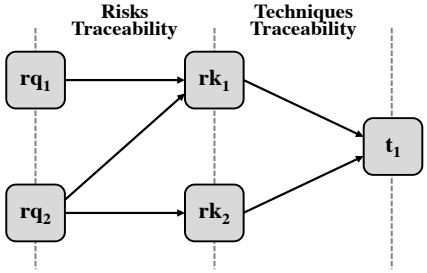

Figure 2. Traceability of risks and mitigation techniques.

Let be two requirements rq_1 and rq_2. Consider that requirement rq_1 is associated to risk rk_1, while requirement rq_2 is associated to both risks rk_1 and rk_2. Also, consider that both risks are dealt by technique t_1. If both requirements are selected, the cost of the technique t_1 is disproportionally partitioned among the requirements, being one third for requirement rq_1 and two thirds for requirement rq_2. This occurs because there are three paths from the requirements rq_1 and rq_2 in direction to the mitigation technique t_1, one from requirement rq_1 and two from requirement rq_2. Thus, the risk management cost rkc_i for each requirement rq_i is calculated as defined in Equation 23 and 24, in which the terms $tp_{j,k}$ represent the probability of applying the technique t_k to the risk rk_j (Equation 21) and the terms tc_k represent the cost of applying technique t_k.

$$rkc_1 = tp_{1,1} \cdot (tc_1/3) \tag{23}$$

$$rkc_2 = tp_{1,1} \cdot (tc_1/3) + tp_{2,1} \cdot (tc_1/3) \tag{24}$$

Based on an *integrated cost-satisfaction evaluation*, recommended in the proposed approach, note that the requirements subset is considered an adequate solution when it maximizes the satisfaction function S (Equation 15) and minimizes the cost function C (Equation 17). The results obtained from both functions allow the cost-satisfaction evaluation, in which an exhaustive or metaheuristic-based algorithm can evaluate and recommend good-enough or even optimal solutions.

3. RELATED WORK

Among the related work, the Bagnall *et al*. proposal [4] is a noteworthy one because it introduces the next release problem and besides, it also introduces the concept of requirements dependencies as an acyclic graph, denoting the requirements and their prerequisites. This dependency relation is defined as transitive. In other words, if a requirement rq_a is dependent on another rq_b, which in turn is dependent on another rq_c, then rq_a is also dependent on rq_c. Despite the importance of requirement dependencies, the approach proposed herein focuses mainly on incorporating a risk-based analysis. However, considering existing proposals that regard dependencies [4] [5], it is not difficult to evolve the proposed approach for including requirements dependencies as part of the evaluation of the requirements costs and clients' satisfaction.

On the same direction of the proposed approach, Ruhe and Greer [6] introduce an iterative model dealing with software risks, in which a set of various releases *m* is recommended. However, in a way different from the proposed approach, this proposal deals with risks as a constraint, defining a limit level that should not be exceeded, as indicated in Equation 25. Thus, unlike the approach proposed herein, risks in this proposal do not impact directly on requirements costs or clients' satisfaction.

$$\sum_{r(i) \in Inc(m)} risk(r_i, R^m) \leq Risk^m \tag{25}$$

In [8], an iterative risk analysis approach for the next release problem is also presented. In this proposal, it assumes that the most critical risks should be delivered on the earlier releases of the software product, but it does not go into detail regarding the values calculated for risks. Each risk is represented in the interval $[1, 5] \in \mathbb{N}$. A penalty is applied when critical risks are selected in later iterations as seen in Equation 26. It defines the term $risk_r$ for representing the risk associated to requirement r, and the term x_i for representing the iteration number of the release, which can assume values in the interval $[1, n] \in \mathbb{N}$. Therefore, similar to the proposed approach, if a critical-risk requirement is selected on a later version, its evaluation becomes progressively worse. However, differently, it deals with risks as a constraint, but not as a factor that impacts on costs and satisfaction.

$$\sum_{r \in R} (risk_r \cdot x_i) \tag{26}$$

In Li *et al*. proposal [7], risk is dealt as a probability of exceeding the budget by a defined margin, with values inversely proportional to the total cost. Differently, in the approach proposed herein, the risk management costs are more accurately obtained from the risk planning stage, in which the costs associated to mitigation techniques must be estimated by the development team.

Yang, Jones and Yang [19] integrates a set of pre-existing software components in a single system. The main challenge is to define the components that provide the lowest risk levels, but provide the best performance for each expected functionality of the software product. It defines the risk level as the product between risk probability and severity. Considering that components are already implemented, risk probability and severity are estimated in a vague way based on code inspection and application context, instead of during the risk planning stage, as proposed herein.

Some traditional requirements prioritization techniques [23] [24] can also be contrasted against the proposed approach. On the one hand, for instance, analytical hierarchical processes evaluate each pair of requirements in order to set the relative importance between each requirement in comparison to other ones. In such approaches, the main disadvantage is the increase in the number of paired evaluations, which have an exponential growth in relation to the number of requirements.

Simpler methods, such as ranking or grouping requirements in different categories according to its importance, can be used in scenarios with a larger number of requirements without too much effort. However, in such cases, the accuracy of data could be hindered. Besides, client prioritization based on traditional negotiation also implies on the adoption of either consensus or vote by majority. The first one comes with the disadvantage of becoming progressively harder as the number of involved requirements and stakeholders increases. The second one can possibly segregate a considerable number of stakeholders, impacting in their satisfaction level in relation to the software product under development.

On the other hand, the approach proposed herein takes in consideration the evaluation of requirements for all clients, in

which their relative importance can be adjusted by the project manager or development team, but never completely discarded, once it does not make sense a client to be disregarded.

4. A PILOT USE CASE

In order to evaluate the applicability of the proposed approach, a prototype implementation has been developed in Java as an exhaustive search algorithm, which assesses all possible solutions defined by all candidate requirements. Note that the search space has an exponential growth, that is, considering a set of *n* candidate requirements, the search space has 2^n solutions.

The prototype implementation has been adopted for conducting an evaluation with a pilot use case, composed by a reduced subset of requirements from a Motorola project [9]. Originally, the project dataset provides 35 requirements, each one with an associated cost. However, considering that 35 requirements defines a large search space that has around 34 billion (2^{35}) candidate solutions, the prototype implementation would take around 1165 days to completely evaluate all those solutions. Thus, for simplicity, the pilot use case is based on two experiments, which adopt 5 and 10 requirements from the original project, defining a small and medium search spaces with 32 and 1024 solutions, respectively. Due to space limits, only input data for the small experiment is detailed, but those for the other one can be supposed by analogy.

Table 2 displays the subset of 5 requirements together with their associated costs and risks. Note that, for each identified requirement, the set of associated risks represents in the proposed approach the term $rt_{i,j}$, which denotes the traceability between the requirement rq_i and the risk rk_j, as defined in Equation 11.

Table 2. Candidate requirements.

Requirement (rq_i)	rq_1	rq_2	rq_3	rq_4	rq_5
Requirement cost (rqc_i)	10	40	80	400	1100
Associated risks ($rt_{i,j}$)	rk_1, rk_2	rk_2	rk_1, rk_3	rk_2, rk_4	rk_1, rk_3, rk_5

For all identified risks, as illustrated in Table 3, it is required to define their respective probability and severity, as well as their associated mitigation techniques. Again, note that, for each identified risk, the set of associated mitigation techniques represents in the proposed approach the term $tt_{j,k}$, which denotes the traceability between the risk rk_j and the mitigation technique t_k, as defined in Equation 14. Although the adopted dataset does not contain risks, a survey that include such information was used as a guideline [25] in order to estimate synthetic data.

Table 3. Software risks and associated techniques.

Risk (rk_j)	Risk Probability (rkp_j)	Risk Severity (rks_j)	Associated Mitigation Techniques ($tt_{j,k}$)
rk_1	Low	Low	t_1, t_6
rk_2	Low	Medium	t_2, t_3
rk_3	Medium	High	t_2, t_5
rk_4	High	High	t_1, t_2, t_4
rk_5	Very high	Very high	t_1, t_2, t_5

Thereafter, for all identified mitigation techniques, as illustrated in Table 4, it is necessary to provide their associated costs, together with classifications as preventive or corrective techniques.

Table 4. Risk mitigation techniques.

Mitigation Technique (t_k)	t_1	t_2	t_3	t_4	t_5	t_6
Technique Cost (tc_k)	10	30	230	15	50	180
Preventive				x	x	x
Corrective	x	x	x			

Besides that, the proposed approach also requires the development team to gather information regarding clients preferences on candidate requirements and their importance from the point of view of the development organization. Table 5 provides such information for all clients and requirements. Note that, although the adopted dataset contains some kind of preference on requirements, such an information is not decomposed into preferences related to individual clients. Due to that, other datasets that include such data were used as a template [26].

Table 5. Clients preferences and importance.

Client (u_i)	Importance (e_i)	Client Preference ($s_{i,j}$)				
		rq_1	rq_2	rq_3	rq_4	rq_5
u_1	1.0	0.2	0.4	0.8	0.4	1.0
u_2	0.7	0.1	0.5	0.2	0.9	0.4
u_3	0.5	0.3	0.3	1.0	0.6	0.6
u_4	0.3	0.8	0.6	0.4	0.4	0.3

From every collected data, it is now possible to apply the proposed approach in order to evaluate the cost and satisfaction levels for each possible subset of selected requirements. The evaluation results for both experiments (5 and 10 candidate requirements) are depicted in Figure 3 and Figure 4. It is important to emphasize that such results simulate an exhaustive search with every possible combination for the set of 5 and 10 candidate requirements.

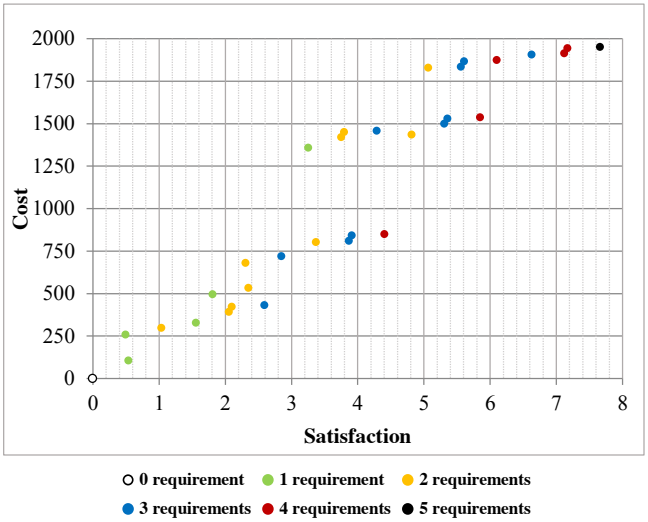

Figure 3. Cost-satisfaction evaluation for 5 requirements.

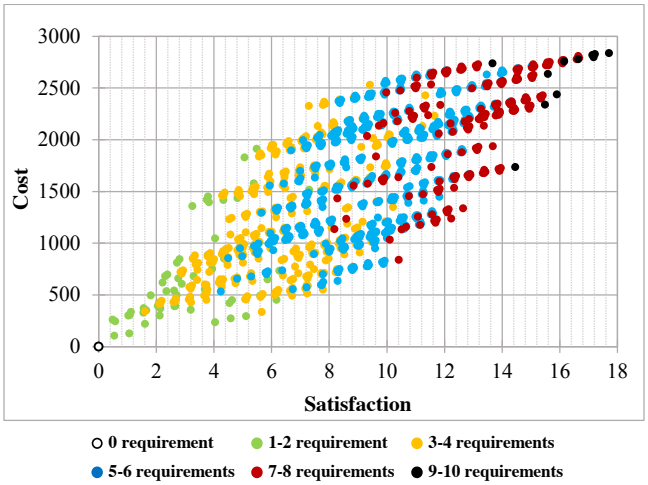

Figure 4. Cost-satisfaction evaluation for 10 requirements.

Based on such pilot results, it was possible to evaluate the applicability of the proposed approach, concluding that all inputs required by the proposed approach are not difficult to obtain from requirements engineering and risk management processes. Besides, the results allow to evaluate whether or not the proposed approach acts according to the expected behavior.

Results in both experiments evince that, as more requirements are progressively selected, there is a stronger tendency for sharing risks among requirements, alleviating the costs penalties introduced by risk mitigation techniques. Such a behavior has the potential to make progressively cheaper the individual cost for each requirement. For instance, consider a solution that selects only the requirement rq_1, associated with risks rk_1 and rk_2, which in turn together are associated with mitigation techniques t_1, t_2, t_3 and t_6. In this case, individually, rq_1 has a cost level of 257.5, as the costs of the mitigation techniques t_1, t_2, t_3 and t_6 are integrally incorporated in the cost level of the solution. Now, as another example, consider a solution that selects all requirements, which is associated with all risks and so all mitigation techniques. In this case, rq_1 contributes with a reduced cost level of 90.738. This reduction occurs because the other requirements also share the risks rk_1 and rk_2, and so also share the mitigation techniques t_1, t_2, t_3 and t_6. Consequently, the costs of the mitigation techniques are shared among all requirements, expressively reducing the contribution of rq_1 in the total cost level of the solution.

As an additional outcome, results show that, in recommended solutions that have requirements with associated higher risk severity, the impact of such risks in the clients' satisfaction is more intense. For instance, consider the requirement rq_2, which is only associated with risk rk_2 that has a medium severity. In this case, based in the clients preference only, the satisfaction level is 1.08. However, including risks severity, the final satisfaction level decays to just 0.54. Now, as another example, consider the requirement rq_5, which is associated with risks rk_1, rk_3 and rk_5 that have low, high and very high severities, respectively. In this case, based in the clients preference only, the satisfaction level is 1.67. However, including risks severity, the final satisfaction level rises to 3.2565.

Another interesting observed outcome that can be noted is that, if selected requirements possess a low implementation cost, the final cost function could still suffer a considerable penalty if such requirements also possesses a great associated risk probability.

The set of obtained results are subject to final evaluation and selection by the project manager. Note that, while in other approaches, such as [9], the development team estimates the cost for each requirement, but the main challenge is to conciliate cost and satisfaction, constrained by a given budget limit per release, avoiding a few critical high-cost requirements or a lot of trivial low-cost requirements that could be crucial to the success of the project. Note that the proposed approach is not constrained by a budget limit. Thus, the approach allows the project manager to analyze how high-cost requirements, such as rq_4 and rq_5 in Table 2, behave in a cost-satisfaction analysis in the solution space.

In summary, the pilot outcomes also reveal that it sounds interesting the concept of introducing risk analysis in requirements costs and clients' satisfaction. For instance, on the one hand, in traditional approaches that do not consider risk analysis, the total cost of selecting all candidate requirements is 1630. On the other hand, in the proposed approach, such a total cost is around 1952, which is more precise and realistic due the inclusion of the costs related to the adoption of a risk management process.

5. CONCLUDING REMARKS

Comparing the proposed approach against related work, it has been characterized that the main contribution of the approach proposed herein is to deal with risk analysis as a measure that impacts on requirements costs and clients' satisfaction, capable of offering value-added information to decision makers. Differently, related work deals with risk analysis as an additional constraint, defining risks as a limit level that should not be exceeded or another evaluation function derived from requirements costs.

The main threats to validity are related to the subjectivity of the input data due to human evaluation. However, such a bias is a usual practice in real projects. As another threat, it can be appointed the unavailability of a real dataset that provides all input data. This could improve the confidence in results.

Despite the relevant contribution and interesting outcomes, the proposed approach needs to be more intensively evaluated. In such a direction, in a first future work, the entire Motorola dataset with all 35 requirements [9] will be evaluated. Complementarily, it is important to identify and evaluate more complex software projects with bigger datasets, such as large-scale free software projects.

Besides, the proposed approach also ought to be adapted to a two-objective metaheuristic-based approach, incorporating the *Pareto optimality* evaluation [5]. In such a direction, the proposed approach will be capable of offering a set of recommended solutions delimited within a Pareto front, instead of providing all possible solutions, turning easier and faster the decision-making process. In an initial study, it has been identified a different number of metaheuristics algorithms, including NSGA-II [27], SPEA2 [28] and MOEA/D [29]. Such metaheuristic search is required to evaluate a dataset with a colossal number of requirements, which increases computational complexity for finding good enough solutions in large, complex search spaces.

6. REFERENCES

[1] C. Larman and V. R. Basili, "Iterative and incremental developments. a brief history," *Computer*, vol. 36, no. 6, p. 10, June 2003.

[2] W. Scacchi, "Process Models in Software Engineering," in *Encyclopedia of Software Engineering*., 2002, p. 24.

[3] Joachim Karlsson and Kevin Ryan, "A Cost-Value Approach for Prioritizing Requirements," *IEEE Software*, vol. 14, no. 5, pp. 67-74, Sep. 1997.

[4] A. J. Bagnall, V. J. Rayward-Smith, and I. M. Whittley, "The next release problem," *Information and Software Technology*, vol. 43, no. 14, pp. 883-890, 2001.

[5] J. J. Durillo, Y. Zhang, E. Alba, M. Harman, and A. Nebro, "A study of the bi-objective next release problem," *Empirical Software Engineering*, vol. 16, no. 1, pp. 29-60, 2011.

[6] G. Ruhe and D. Greer, "Quantitative studies in software release planning under risk and resource constraints," in *Proceedings of the 2003 International Symposium on Empirical Software Engineering*, Washington, DC, USA, 2003, pp. 262-.

[7] L. Li, M. Harman, E. Letier, and Y. Zhang, "Robust next release problem: Handling uncertainty during optimization," in *Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation*, New York, NY, USA, 2014, pp. 1247-1254.

[8] F. Colares, J. Souza, R. Carmo, C. Pádua, and G. R. Mateus, "A new approach to the software release planning," in *Software Engineering, 2009. SBES '09. XXIII Brazilian Symposium*, Out. 2009, pp. 207-215.

[9] P. Baker, M. Harman, K. Steinhofel, and A. Skaliotis, "Search based approaches to component selection and prioritization for the next release problem," in *Software Maintenance, 2006. ICSM '06. 22nd IEEE International Conference*, Set. 2006.

[10] Imran Alam, "Role of Software Metrices in identifying the risk of project," *International Journal of Advancement in Engineering Technology, Management & Applied Science*, vol. 1, no. 1, p. 7, June 2014.

[11] J. Verner, J. Sampson, and N. Cerpa, "What factors lead to software project failure?," in *Research Challenges in Information Science, 2008. RCIS 2008. Second International Conference*, Junho 2008, pp. 71-80.

[12] S.P. Masticola, "A simple estimate of the cost of software project failures and the breakeven effectiveness of project risk management," in *Proceedings of the First International Workshop on The Economics of Software and Computation, ESC '07*, Washington, DC, USA, 2007, pp. 6--.

[13] B. W. Boehm, "Software risk management: principles and practices," *IEEE Software*, vol. 8, no. 1, pp. 32-41, Jan. 1991.

[14] S. Islam and S. H. Houmb, "Integrating risk management activities into requirements engineering," in *Research Challenges in Information Science (RCIS), 2010 Fourth International Conference on,* Nice, France, 2010, pp. 299-310.

[15] Prasad Rajagopal , Roger Lee , Thomas Ahlswede, Chia-Chu Chiang, and Dale Karolak, "A New Approach for Software Requirements Elicitation," in *Sixth International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing*, 2005.

[16] Tharwon Arnuphaptrairong, "Top Ten Lists of Software Project Risks: Evidence from the Literature Survey," in *International MultiConference of Engineers and Computer Scientists (IMECS 2011)*, vol. I, Hong Kong, 2011.

[17] Tom DeMarco and Tim Lister, "Risk Management during Requirements," *IEEE Software*, vol. 20, no. 5, pp. 99-101, Oct. 2003.

[18] Mohd Huma Hayat Khan and Suriayati bt Chuprat Naz'ri bin Mahrin, "Factors Generating Risks during Requirement Engineering Process in Global Software Development Environment," *International Journal of Digital Information and Wireless Communications (IJDIWC)*, vol. I, no. 4, pp. 63-78, 2014.

[19] L. Yang, B. Jones, and S. Yang, "Genetic algorithm based software integration with minimum software risk," *Information and Software Technology*, vol. 48, no. 3, pp. 133-141, 2006.

[20] M. S. Feather and S. L. Cornford, "Quantitative risk-based requirements reasoning," *Requir. Eng.*, vol. 8, pp. 248-265, Nov. 2003.

[21] Paul L. Bannerman, "Risk and risk management in software projects: A reassessment," *Journal of Systems and Software*, vol. 81, no. 12, pp. 2118-2133, Dec. 2008.

[22] Hongliang Zhang, "A redefinition of the project risk process: Using vulnerability to open up the event-consequence link," *International Journal of Project Management*, vol. 25, no. 7, pp. 694-701, Oct. 2007.

[23] Patrik Berander and Anneliese Andrews, "Requirements Prioritization," in *Engineering and Managing Software Requirements*. Berlin, Germany: Springer, 2005, ch. 4, pp. 69-94.

[24] J. M. Fernandes and R.J. Machado, "Requirements Negotiation and Prioritisation," in *Requirements in Engineering Projects*. Switzerland: Springer, 2016, ch. 6, pp. 119-134.

[25] S. M. Neves, C. E. S. da Silva, V. A. P. Salomon, A. F. da Silva, and B. E. P. Sotomonte, "Risk management in software projects through Knowledge Management techniques: Cases in Brazilian Incubated Technology-Based Firms," *International Journal of Project Management*, vol. 32, no. 1, pp. 125-138, 2014.

[26] M. R. Karim and G. Ruhe, "Bi-objective Genetic Search for Release Planning in Support of Themes," in *International Symposium on Search Based Software Engineering (SSBSE 2014)*, Fortaleza, Brazil, 2014, pp. 123-137.

[27] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: Nsga-ii," *Trans. Evol. Comp*, vol. 6, pp. 182-197, Apr. 2002.

[28] E. Zitzler, M. Laumanns, and L. Thiele, "Spea2: Improving the strength pareto evolutionary algorithm," 2001.

[29] Q. Zhang and H. Li, "Moea/d: A multiobjective evolutionary algorithm based on decomposition," *IEEE Transactions on Evolutionary Computation*, vol. 11, pp. 712-731, Dec. 2007.