# A Multi-Objective, Risk-based Approach for Selecting Software Requirements

Aruan G. Amaral and Gledson Elias
*Informatics Center, Federal University of Paraíba, João Pessoa, Brazil*

Keywords: Multi-Objective Evolutionary Computing, Software Requirements, Risk Management.

Abstract: In iterative and incremental development approaches, there is great interest in delivering system releases on-budget, but raising stakeholders' satisfaction as much as possible. In the field of Search Based Software Engineering (SBSE), such a problem is known as the Next Release Problem (NRP), which is handled in existing proposals by reformulating the requirements selection process as an optimization problem solved by metaheuristics, providing a set of recommendations with the highest customers' satisfactions as well as the lowest development costs. Despite their contributions, most of current proposals do not address software risks, which represent a key aspect that can deeply impact on project cost and stakeholders' satisfaction. In such a direction, this paper proposes a multi-objective, risk-based approach for the NRP problem, in which a risk analysis is incorporated to estimate the impact of software risks in development cost and stakeholders' satisfaction. Experimental results reveal the efficiency and practical applicability of the proposed approach.

## 1 INTRODUCTION

In software project planning and management, budget constraints apply on every iteration of all software projects (Scacchi, 2001). Thus, in order to deliver a system release on-budget when the total cost of candidate requirements exceeds the available budget, project managers face the problem of deciding on which requirements should be prioritized for the next release. Besides the effort on negotiating such requirements with stakeholders, it is also required to balance trade-offs among critical aspects, such as budget, requirements costs, customers' preferences and their importance, keeping costs under control and raising the satisfaction for all stakeholders.

In such a scenario, the software requirements selection process represents a complex, challenging and error-prone task, in which the adoption of manual, ad-hoc approaches are impractical due to the large amount of correlated data and conflicts of interest among stakeholders. Besides, the effort for conciliating and balancing trade-offs turns out to be harder as the set of requirements becomes larger.

As a mean to lighten the complexity, effort and mistakes, information related to requirements and stakeholders must be made computable by automated, systematic decision-making approaches. As one of the first approaches, the cost-importance model (Karlsson and Ryan, 1997) aims to minimize costs and delivery time, as well as to maximize the satisfaction level perceived by stakeholders.

Later, the software requirements selection process was represented as an optimization problem known as the *Next Release Problem* (NRP) (Bagnall *et al.*, 2001). In the SBSE field, several proposals characterize and solve the NRP problem from different and complimentary viewpoints (Huhe and Greer, 2003; Baker *et al.*, 2006; Colares *et al.*, 2009; Durillo *et al.*, 2011a; Li *et al.*, 2014), evolving from a single-objective to a multi-objective perspective. In the former, proposals focus on finding the better solution that keeps project budget under control and raises stakeholders' satisfaction. In the latter, a pre-allocated budget constraint is discarded in favor of offering not only a single solution but a set of good solutions, providing different recommendations that produce the highest stakeholders' satisfaction as well as the lowest development cost, known as the Pareto Front, in which the set of solutions cannot be improved in any dimension without degradation in another (Zitzler and Thiele, 1999).

Despite contributions, such proposals do not address software risks, which can deeply impact on requirements costs and stakeholders' satisfaction.

Amaral, A. and Elias, G.
A Multi-Objective, Risk-based Approach for Selecting Software Requirements.
DOI: 10.5220/0006555503380346
In *Proceedings of the 10th International Conference on Agents and Artificial Intelligence (ICAART 2018) - Volume 2*, pages 338-346
ISBN: 978-989-758-275-2

Hence, the integration of risk analysis sounds to be an insightful contribution to the NRP formulation, which can be reinforced by the fact that the adoption or lack of risk management is a key reason for software project success (Alam, 2014) or failure (Verner *et al.*, 2008), respectively.

In such a direction, this paper proposes a multi-objective, risk-based approach for the NRP problem, called **SR²** (*Selection of Requirements based on Software Risks*), in which a risk analysis is incorporated for estimating the impact of risks on requirements costs and stakeholders' satisfaction. In *SR²*, candidate requirements are associated to identified risks, which in turn are related to risk mitigation techniques. Then, based on risks probability and severity, together with the cost of applying mitigation techniques, *SR²* estimates the impact of risks on both requirements costs and stakeholders' satisfaction. By exploring the widely adopted multi-objective optimization algorithm NSGA-II, experimental results based on two semi-real datasets reveal the potential efficiency and practical applicability of the proposed approach.

The remainder of this paper is organized as follows. Section 2 presents fundaments related to multi-objective optimization. Section 3 presents *SR²* in detail. In Section 4, a case study with two datasets is presented. Section 5 discusses related work. Then, Section 6 presents final reflexions and future work.

## 2 FUNDAMENTS

For a Multi-objective Optimization Problem (MOP), let $\vec{x}^* = [x_1^*, x_2^*, \dots, x_n^*]$ be a vector of decision variables that **minimizes** the vector of objective functions $\vec{f}(\vec{x}) = [f_1(\vec{x}), f_2(\vec{x}), \dots, f_k(\vec{x})]^T$, in which $\vec{x} = [x_1, x_2, \dots, x_n]^T$ is the vector of decision variables. Any point $\vec{x}$ defines a solution and the set of all solutions shapes the search space $\Omega$ (Durillo *et al.*, 2011a). The main goal of a MOP relies on finding a set of good enough or even optimal solutions that minimize the objective functions $\vec{f}(\vec{x})$. In MOPs, **Pareto dominance** is defined by comparing a given solution in relation to all others found so far. Thus, a solution $\vec{x}'$ is said to dominate a solution $\vec{x}''$, denoted by $\vec{x}' \preccurlyeq \vec{x}''$, if and only if $\forall_{i=1,2\cdots,k}(f_i(\vec{x}') \leq f_i(\vec{x}''))$ and $\exists_{i=1,2,\cdots,k}(f_i(\vec{x}') < f_i(\vec{x}''))$, meaning that $\vec{x}'$ has a lower or equal objective score for all objective functions, but there is at least one in which $\vec{x}'$ has a lower score in contrast to $\vec{x}''$.

The set of solutions that are non-dominated by any other solution in the whole search space $\Omega$ is defined as the **Pareto Optimal Set**, represented by $P^* = \{\vec{x} \in \Omega \mid \nexists \vec{x}' \in \Omega, \vec{x}' \preccurlyeq \vec{x}\}$. Thus, Pareto optimal solutions are non-dominated solutions. In MOPs, the search for the optimal set $P^*$ is the main goal, which is represented as a **Pareto Front**, defined as the set $PF^* = \{\vec{f}(\vec{x}) \mid \vec{x} \in P^*\}$. In large search spaces, an approximation of the optimal set $P^*$ is desired.

Besides this optimality feature, solutions should not be concentrated on a single region of the search space or too scattered. A uniform spread among the Pareto front is a desirable quality for multi-objective optimization algorithms (Durillo *et al.*, 2011a). In this regard, two quality indicators can be used to measure both desirable features for a set of good solutions: **Hypervolume** and **Spread**. Both quality indicators require the optimal set $P^*$, but due to computational complexity it is not always possible to find such a set. In such cases, a **reference front** with best-known solutions is a usual approximation.

Hypervolume $(HV)$ evaluates solutions convergence and distribution in relation to the optimal front (Zitzler and Thiele, 1999). Considering a set of non-dominated solutions $Q = \{\vec{x}^1, \vec{x}^2, \dots, \vec{x}^n\}$, $HV(Q)$ can be estimated by building hypercubes using objective function values of each solution as coordinates in relation to a reference point $\vec{x}^w$, which is a vector built with the worst possible solution for each objective. Eq. 1 provides the hypervolume expression, in which each hypercube $v_i$ has the reference point $\vec{x}^w$ and the solution $\vec{x}^i$ as the diagonal corners of the hypercube.

$$HV(Q) = volume\left(\bigcup_{i=1}^{|Q|} v_i\right) \tag{1}$$

In turn, the normalized hypervolume $(HVR)$ is defined as $HVR = HV(Q)/HV(P^*)$, reaching a maximum value of one when the non-dominated solutions $Q$ nears the optimal set $P^*$ (Deb, 2001). In contrast, a low value means either solutions far from $P^*$ or clustered in a small region of the search space.

Spread $(\Delta)$ denotes how well non-dominated solutions $Q = \{\vec{x}^1, \vec{x}^2, \dots, \vec{x}^n\}$ are distributed among the Pareto Front (Deb, 2001), as defined in Eq. 2. The terms $d_f$ and $d_l$ define the Euclidian distance between the first/last optimal solutions in $P^*$ and the first/last solutions $\vec{x}^1$ and $\vec{x}^n$ in $Q$. In between, the terms $d_i$ represent the Euclidian distance between all consecutive solutions $\vec{x}^i$ and $\vec{x}^{i+1}$ in $Q$, which define $N-1$ intervals. Finally, the term $\bar{d}$ denotes the mean distance between all consecutive solutions in $Q$. A spread value near zero means that the first/last points in $Q$ are near the respective points in $P^*$ and the distances between consecutive solutions in $Q$ are near $\bar{d}$, reaching a near-perfect distribution.

$$\Delta = \frac{\left(d_f + d_l + \sum_{i=1}^{N-1} |d_i - \bar{d}|\right)}{\left(d_f + d_l + (N-1) \cdot \bar{d}\right)} \tag{2}$$

## 3 PROPOSED APPROACH

In the NRP context, as shown in Fig. 1, *SR*$^2$ is a three-layered, automated, multi-objective, risk-based approach for selecting software requirements, in which a software risk analysis estimates risks impact on development cost and stakeholders' satisfaction.

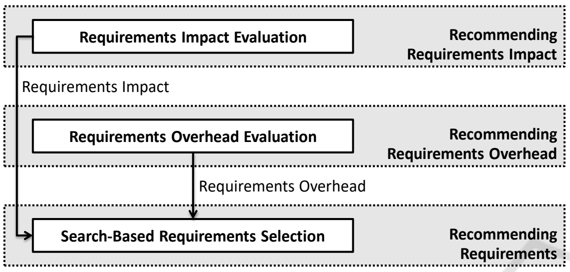
Requirements Impact Evaluation
Recommending Requirements Impact
Requirements Impact
Requirements Overhead Evaluation
Recommending Requirements Overhead
Requirements Overhead
Search-Based Requirements Selection
Recommending Requirements

Figure 1: Three-layered architecture of the SR$^2$ approach.

### 3.1 Requirements Impact

The phase *Recommending Requirements Impact* estimates a penalty to be applied to each software requirement when guessing customers' satisfaction. To do that, it is assumed that software risks associated to requirements imply an impact on their implementation. In *SR*$^2$, **software requirements** are represented by the set $RQ = \{rq_1, rq_2, \dots, rq_n\}$, where each $rq_i$ denotes a candidate requirement for the next release. In turn, **software risks** are denoted by the set $RK = \{rk_1, rk_2, \dots, rk_m\}$, where each $rk_j$ represents a risk that can arise during the project.

Usually, a risk is defined as a material or financial loss, or any other event that must be avoided (Boehm, 1991). Every risk is associated with *severity* and *probability* values, which indicates respectively the negative consequences of the risk event and the likelihood of such undesirable event.

Considering the difficulty of associating accurate values, a fuzzy-driven notation is adopted. As shown in Table 1, linguistic terms classify probability and severity as discrete numerical values according a five-point Likert scale (Bannerman, 2008), whose values are in $FT = \{0.05, 0.25, 0.50, 0.75, 0.95\}$.

Table 1: Risk probability and severity.

| Term | very low | low | medium | high | very high |
|---|---|---|---|---|---|
| Value | 0.05 | 0.25 | 0.50 | 0.75 | 0.95 |

In *SR*$^2$, the risks probability is represented by the set $RKP = \{\, rkp_j \mid \exists rk_j \in RK,\ rkp_j \in FT \,\}$ and in turn, the risks severity is characterized by the set $RKS = \{\, rks_j \mid \exists rk_j \in RK,\ rks_j \in FT \,\}$.

As a mean to represent **risks traceability**, each requirement $rq_i$ might be associated with zero, one or several risks $rk_j$. Thus, the relationship defined by $RT_{RQ,RK} = \{rt_{i,j} \mid \exists rq_i \in RQ,\ \exists rk_j \in RK,\ rt_{i,j} \in \{0,1\}\}$ characterizes such a traceability, in which $rt_{i,j}$ denotes the traceability between requirement $rq_i$ and risk $rk_j$, assuming a value of one or zero to indicate its existence or not, respectively.

Based on every involved concept, the **requirements impact** is represented by the set $IMP = \{\, imp_i \mid \exists rq_i \in RQ,\ imp_i \in \mathbb{R} \,\}$, where each term $imp_i$ can be estimated by Eq. 3, representing the impact of all risks on requirement $rq_i$. Note that Eq. 3 establishes the relation among the following terms: *(i)* the traceability $rt_{i,j}$ among requirement $rq_i$ and its associated risks $rk_j$; and *(ii)* the risk severity $rks_j$ associated to traced risks $rk_j$.

$$imp_i = \sum_{rk_j \in RK} rks_j \cdot rt_{i,j} \tag{3}$$

### 3.2 Requirements Overhead

Then, the phase *Recommending Requirements Overhead* estimates a penalty to be applied to each software requirement when assessing development cost. To do that, it is assumed that mitigation techniques adopted for reducing/eliminating risks consequences lead to an overhead on requirements implementation. In *SR*$^2$, **mitigation techniques** are represented by the set $T = \{t_1, t_2, \dots, t_q\}$, each one possessing an associated cost. **Technique costs** are denoted by the set $TC = \{tc_k \mid \exists t_k \in T,\ tc_k > 0\}$.

Each mitigation technique can help in mitigating one or more software risks. Such a **technique traceability** is represented in *SR*$^2$ by the relationship $TT_{RK,T} = \{tt_{j,k} \mid \exists rk_j \in RK, \exists t_k \in T, tt_{j,k} \in \{0,1\}\}$. Each $tt_{j,k}$ associates risk $rk_j$ to technique $t_k$, assuming values equal to zero/one to indicate the absence/presence of the association. Mitigation techniques are classified as **preventive** and **corrective**. In the former, it attempts to avoid risk occurrence, being applied regardless of the risk event happens or not. In the latter, it attempts to mitigate or eliminate risk consequence, being applied after occurring the risk event.

Based on such related concepts, **requirements overhead** is represented in *SR*$^2$ by $OVR_{RQ,RK,T} = \{\, ovr_{i,j,k} \mid \exists rq_i \in RQ, \exists rk_j \in RK, \exists t_k \in T, ovr_{i,j,k} \in \mathbb{R} \,\}$, where each term $ovr_{i,j,k}$ can be estimated by Eq. 4, representing the overhead of technique $t_k$ related to risk $r_j$, which in turn is related to requirement $rq_i$. Note that Eq. 4 estimates the overhead based on the cost $tc_k$ of applying technique $t_k$ factored by the probability $tp_{j,k}$ of applying technique $t_k$ for avoiding or mitigating risk $rk_j$. In turn, the probability $tp_{j,k}$ is given by Eq. 5, in which the cost for preventive techniques is fully considered as they are applied regardless of risk events happen or not, and the cost for corrective techniques depends on risk probabilities ($rkp_j$) as they are not applied until the uncertain occurrence of risk events.

$$ovr_{i,j,k} = rt_{i,j} \cdot tt_{j,k} \cdot tc_k \cdot tp_{j,k} \tag{4}$$

$$tp_{j,k} = \begin{cases} 1 & if\ \boldsymbol{preventive}\ technique, \\ rkp_j & if\ \boldsymbol{corrective}\ technique \end{cases} \tag{5}$$

## 3.3 Requirements Recommendation

Now, based on the multi-objective genetic algorithm NSGA-II (Deb *et al.*, 2002), the phase *Recommending Requirements* selects requirements, in which risks impose an impact on customers' satisfaction and an overhead on requirements costs, producing a set of recommendations for the software requirements to be implemented in the next. release.

Due to space limit, the following discussion does not detail NSGA-II, but has the focus on the evaluation of candidate solutions, based on two fitness functions that assess customers' satisfaction and requirements cost, as shown in Fig. 2.

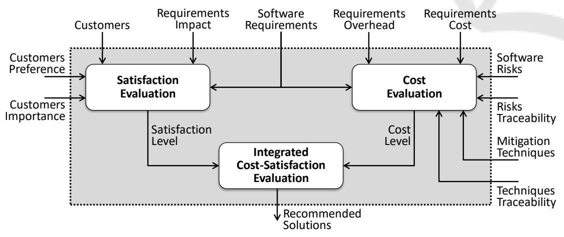

Figure 2: Recommending Requirements.

In the *SR²* proposal, **customers** are represented by the set $U = \{u_1, u_2, \dots, u_p\}$, each one defining a set of requirements preferences, called **customers preference**, which is denoted by the relationship $S_{U,RQ} = \{s_{l,i} \mid \exists u_l \in U,\ \exists rq_i \in RQ\ ,\ s_{l,i} \in [0,1]\}$, where each term $s_{l,i}$ can assume values in the interval $[0,1] \in \mathbb{R}$. In order to deal with conflicts of interest among customers, *SR²* adopts the concept of c**ustomers importance**, which is denoted by the set $E = \{\, e_l \mid \exists u_l \in U,\ e_l \in (0,1]\,\}$, indicating how important each customer is to the business strategy of the development organization. Each term $e_l$ can assume values in the interval $(0,1] \in \mathbb{R}$.

Besides, in order to evaluate the next release cost, **requirements cost** is represented in *SR²* by the set $RQC = \{rqc_i \mid \exists rq_i \in RQ,\ rqc_i > 0\}$, where each $rqc_i$ represents the development cost associated to requirement $rq_i$ based on estimations produced by the development team.

Finally, it is known that just a subset of candidate requirements $RQ$ will be selected for next release. As such, each possible **recommended solution** is defined in the proposed approach by the set $X = \{\, x_i \mid \exists rq_i \in RQ,\ x_i \in \{0,1\}\,\}$, where each term $x_i$ assumes values equal to one/zero, denoting that requirement $rq_i$ has been chosen or not.

### 3.3.1 Satisfaction Evaluation

The **satisfaction level** ($S_X$) can be estimated by Eq. 6, indicating the satisfaction perceived by customers for candidate solution $X$. It is modelled by the total sum for each customer $u_l$ and requirement $rq_i$, considering the product among the following terms: *(i)* the preference level $s_{l,i}$ that customer $u_l$ has in relation to requirement $rq_i$; *(ii)* the importance level $e_l$ that the development organization assigned to customer $u_l$; *(iii)* the impact $imp_i$ associated to requirement $rq_i$; and *(iv)* the selector $x_i$ that represents the selection or not of requirement $rq_i$.

$$max\ \ S_X = \textstyle\sum_{u_l \in U} \sum_{rq_i \in RQ} s_{l,i} \cdot e_l \cdot imp_i \cdot x_i \tag{6}$$

Note that impact $imp_i$ associated to requirement $rq_i$ is estimated in a way that the higher the severity associated to related risks, the higher the impact. Thus, *SR²* adopts the premise that software processes must first deal with most critical risks as a mean to maximize the chances of the software project be successful, as usually perceived in real projects that focus on most critical risks (Alam, 2014).

### 3.3.2 Cost Evaluation

The **cost level** ($C_X$) can be estimated by Eq. 7, indicating the total cost for implementing a candidate solution $X$. It is modelled by the total sum among the development cost $rqc_i$ and the risk management cost $rkc_i$ for each requirement $rq_i$. As defined, risk management cost represents a penalty in the total cost of the next release, which is a typical strategy in real software projects dealing with risks (Bannerman, 2008). As defined in Eq. 8, risk management cost $rkc_i$ is given by the total sum of risk management cost $rkc_{i,j}$ for each risk $rk_j$ related to requirement $rq_i$.

$$min\ C_X = \sum_{rq_i \in RQ} (rqc_i + rkc_i) \cdot x_i \quad (7)$$

$$rkc_i = \sum_{rk_j \in RK} rkc_{i,j} \quad (8)$$

Remember that a mitigation technique $t_k$ can be associated to one or more risks $rk_j$. Thus, the cost $tc_k$ of applying technique $t_k$ can be shared among all associated risks $rk_j$. In Eq. 9, risk management cost $rkc_{i,j}$ is defined by the ratio between the following terms: *(i)* the total sum of the overhead $ovr_{i,j,k}$ of applying each technique $t_k$ related to risk $rk_j$ and requirement $rq_i$, and *(ii)* the number of times $ta_k$ that technique $t_k$ is applied in all requirements $rq_i$ and risks $rk_j$. Now, considering the term $ta_k$, as defined in Eq. 10, 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 and the technique ($tt_{j,k}$) in question.

$$rkc_{i,j} = \sum_{t_k \in T} ovr_{i,j,k} / ta_k \quad (9)$$

$$ta_k = \sum_{rq_i \in RQ} \sum_{rk_j \in RK} x_i \cdot rt_{i,j} \cdot tt_{j,k} \quad (10)$$

Once satisfaction $S_X$ and cost $C_X$ are estimated for all evaluated solutions, NSGA-II tries to find a good enough set of **recommended solutions**, as much as possible near the Pareto optimality. Note that the problem faced by the development organization is to find a set of recommended solutions that maximizes the satisfaction function $S_X$ (Eq. 6) and minimizes the cost function $C_X$ (Eq. 7).

## 4 RESULTS AND DISCUSSIONS

The proposed approach has been evaluated using two semi-real datasets (Karim and Ruhe, 2014). The first one comprises 25 requirements and 8 customers regarding a *Release Planner* tool. The second one includes 50 requirements and 4 customers regarding a project for *Microsoft Word*. Both datasets make available data concerning requirements cost, customers preference as well as importance. The remainder of the required input data related to risks and mitigation techniques were synthetically estimated based on other information available from datasets. It is important to say that 4 software risks and 4 mitigation techniques were gathered for the first dataset, while 8 software risks and 6 mitigation techniques were deduced for the second dataset.

As already mentioned, *SR*$^2$ adopts the multi-objective optimization algorithm NSGA-II (Deb *et al.*, 2002). The rationale is mainly based on a systematic review (Pitangueira *et al.*, 2015), which points out that NSGA-II is the most used algorithm in multi-objective NRP proposals. Also, it is common to find that competing algorithms use NSGA-II results as a benchmark to validate their own results. NSGA-II adopts an elitist approach, in which non-dominated and diversity solutions are favoured. Note that, although NSGA-II defines the replacement operator of a genetic algorithm, it is still needed to choose mutation, crossover and selection operators. In all experiments, NSGA-II was tuned with flipping mutation with a probability of *1/n*, where $n$ is the number of requirements. Besides, it adopts uniform crossover with a rate of 90% and binary tournament selection.

As a mean to evaluate the quality of the NSGA-II findings, a random search was also adopted to provide a *sanity check* (Harman *et al.*, 2012). To assess search capabilities of both NSGA-II and random search, each experiment performs 100 independent runs, achieving a good confidence in results. Also, the Wilcoxon test was used considering a confidence interval of 95%.

The parametrization of both algorithms is described in Table 2, tuned through successive calibrations tests. As can be seen, the population size for each experiment is quadrupled in relation to the number of candidate requirements. Regarding the number of generations, which is the NSGA-II stop criteria, it is configured considering the number of candidate requirements ($n$) and the size of the search space ($2^n$), calibrated by the expression $5 \cdot n \cdot 2^{n/25}$.

Table 2: Parametrization for experiments.

| | NSGA-II | Random Search |
|---|---|---|
| **Experiment I (25 requirements)** | | |
| **Population** | 100 | - |
| **Generations** | 250 | - |
| **Evaluated Solutions** | 25.000 | 25.000 |
| **Experiment II (50 requirements)** | | |
| **Population** | 200 | - |
| **Generations** | 1.000 | - |
| **Evaluated Solutions** | 200.000 | 200.000 |

Experiments were evaluated using normalized hypervolume ($HVR$) and spread ($\Delta$). Remember that such quality indicators require some knowledge about the Pareto/reference fronts. In *Experiment I*, considering the reduced size of the search space defined by 25 requirements, the Pareto Front was discovered using an exhaustive search, finding 105 non-dominated solutions. In *Experiment II*, it was not possible to perform an exhaustive search due to the large search space defined by 50 requirements. Thus, *Experiment II* adopts just a reference front, obtained from intensively repeated runs of the case, finding 386 non-dominated solutions.

All algorithms (NSGA-II, random search and exhaustive search) have been implemented in Java using the jMetal framework (Durillo and Nebro, 2011b), running on a microcomputer equipped with a Quad-Core Intel i5 2400 processor and 8GB of DDR3 RAM, instantiated with 4 concurrent threads.

Obtained results have been compiled in Table 3. Considering hypervolume in *Experiment I*, NSGA-II provided an excellent value of *9,97e-01*, indicating that solutions are very close to those in the Pareto Front, obtained through an exhaustive search. In contrast, the random search obtained a not-so-good value of *4,51e-01*, indicating that solutions are very far from those in the Pareto Front.

The same observed behaviour holds true for *Experiment II*. However, it can be noted a significant increase in the gap among the hypervolume achieved by NSGA-II and the random search, indicating that even worse results have been found in the random search due to the augmented search space.

Table 3: HVR, spread and execution time.

| | Experiment I | Experiment II |
|---|---|---|
| | **HVR - Mean and standard deviation** | |
| **NSGA-II** | $\mathbf{9,97e-01_{1,7e-02}}$ | $\mathbf{9,98e-01_{1,6e-04}}$ |
| **Random Search** | $4,51e-01_{1,5e-01}$ | $2,83e-01_{1,1e-01}$ |
| | **Spread - Mean and standard deviation** | |
| **NSGA-II** | $\mathbf{3,54e-01_{2,1e-02}}$ | $\mathbf{3,75e-01_{2,2e-02}}$ |
| **Random Search** | $6,74e-01_{8,5e-02}$ | $7,41e-01_{5,1e-02}$ |
| | **Execution time** | |
| **NSGA-II** | **0′09″** | **2′01″** |
| **Random Search** | 0′08″ | **1′59″** |

Regarding the spread indicator, as evinced in Table 3, in both experiments, NSGA-II always outperforms the random search, finding non-dominated solutions with better uniform distribution along the search space.

Considering execution time, it can be perceived in Table 3 that both algorithms have similar performance, which is a positive feature for NSGA-II, since its recommended solutions are much better than those provided by the random search. To emphasize the excellent performance and quality of results provided by NSGA-II, it is important to say that the exhaustive search has taken *38'09''* to find the optimal set, while NSGA-II has taken just *0'09''* to find solutions very close to the Pareto Front.

Fig. 3 and Fig. 4 illustrate the recommended solutions for NSGA-II and the random search, compared to Pareto/reference fronts in experiments I and II, respectively. In both cases, the illustrated results are the best *HVR* value found among all executions. As can be inferred, differently from the random search, NSGA-II has a trend of finding solutions very close to Pareto or reference front.

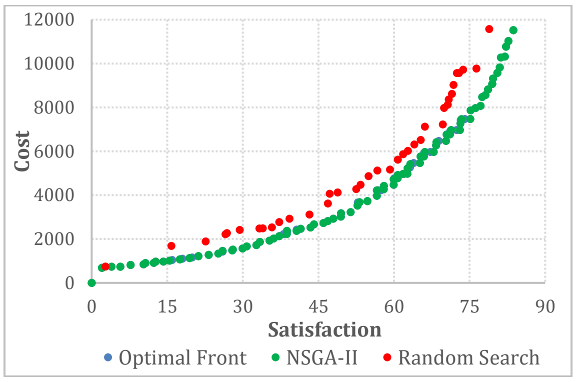

Figure 3: Results for experiment I.

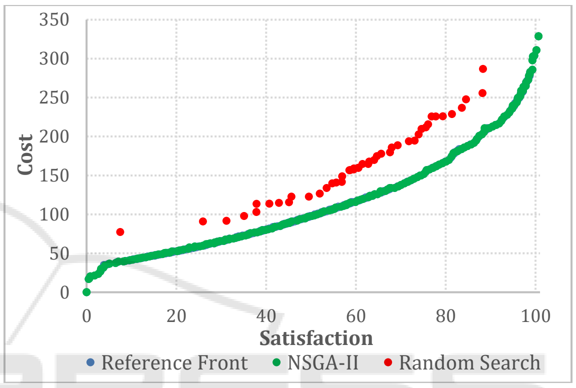

Figure 4: Results for experiment II.

As another quality indicator, Table 4 evinces that NSGA-II obtained a higher number of solutions in the Pareto and reference fronts, respectively. In *Experiment I*, NSGA-II obtained almost the whole population as non-dominated solutions in the Pareto front, which was in average a value of *97,8* solutions. Similarly, in *Experiment II*, NSGA-II also produced a considerable number of non-dominated solutions in the reference front, in average a value of *164* solutions. Note that the random search almost does not found solutions in both fronts.

Table 4: Number of solutions found in fronts.

| | Experiment I | Experiment II |
|---|---|---|
| **NSGA-II** | $\mathbf{9,78e+01_{0,14e+01}}$ | $\mathbf{1,64e+02_{0,54e+01}}$ |
| **Random Search** | $0,06e+00_{0,23e+00}$ | $0,00e+00_{0,00e+00}$ |

## 5 RELATED WORK

SBSE has been successfully applied in many activities throughout the software lifecycle. More recently, NRP proposals have evolved from a single to multi-objective perspective.

Among the related work, the proposal by Bagnall *et al.* (2001) is a noteworthy one because it introduces the NRP problem and launches 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 requirement $rq_a$ is dependent on another $rq_b$, which in turn depends on another $rq_c$, then $rq_a$ also depends on $rq_c$. Despite the importance of requirements dependencies, *SR*$^2$ focuses mainly on incorporating a risk-based analysis. However, considering existing proposals that regard dependencies (Bagnall *et al.*, 2001; Durillo *et al.*, 2011a), it is not difficult to evolve the proposed approach for including dependencies as part of the evaluation of customers' satisfaction and requirements costs.

On the same direction of the proposed approach, Ruhe and Greer (2003) introduce an iterative model dealing with software risks, in which a set of various releases $m$ is recommended. However, unlike the proposed approach, this proposal deals with risks as a constraint, defining a limit level that should not be exceeded. Thus, unlike the approach proposed herein, risks in this proposal do not impact directly on customers' satisfaction or requirements costs.

In (Colares *et al*., 2009), an iterative risk analysis approach for the NRP problem is also presented. It assumes that the most critical risks should be delivered on 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 $[1, 5] \in \mathbb{N}$. Besides, a penalty is applied when critical risks are selected in later iterations. That is, the greater the iteration number, the higher the estimated penalty. Like the *SR*$^2$ proposal, it defines the notion of representing risks associated to requirements. Therefore, similar to the proposed approach, if a critical-risk requirement is selected on a later release, its evaluation becomes progressively worse. However, differently, it deals with risks as a constraint, but not as a factor that impacts on customers' satisfaction and requirements costs. It is important to note that other proposals (Brasil *et al.*, 2011; Saraiva *et al.*, 2016) also adopt a risk function similar to that presented in (Colares *et al*., 2009).

Dantas *et al.* (2015) also address a release planning problem for a multiple number of releases. Differently from previous proposals (Colares *et al.*, 2009; Brasil *et al.*, 2011; Saraiva *et al.*, 2016), the risk associated to a software requirement is directly inserted as a penalty into the objective function, instead of defining a separate objective function for evaluating such risks. However, in a way similar to such proposals, the penalty estimated for risks is also reduced when a critical-risk requirement is allocated into earlier releases. On the one hand, similar to *SR*$^2$, the risks associated to selected requirements directly affect the satisfaction level perceived by customers. This is because the objective function also regards the importance value assigned by customers to all requirements. However, on the other hand, differently from *SR*$^2$, the cost constraint does not reflect the impact of risks.

In the proposal by Li *et al.* (2014), risk is dealt as a probability of exceeding the budget by a defined margin, with values inversely proportional to the total cost. Differently, in *SR*$^2$, the risk management costs are more accurately gathered from the risk management process, in which the costs associated to mitigation techniques must be estimated.

Yang *et al.* (2006) integrate 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 best performance for each expected functionality. It defines the risk level as the product between risk probability and severity, which are estimated based on code inspection and application context, instead of during a risk management analysis, as proposed herein.

## 6 CONCLUDING REMARKS

In this paper, a new approach for the multi-objective NRP problem has been presented, reshaping both the cost and satisfaction objective functions to address risks. Experiments with two semi-real datasets have been presented, in which the proposed approach, exploring the NSGA-II algorithm, has obtained a higher number of recommended solutions closer to Pareto and reference fronts, and besides has also produced non-dominated solutions better distributed.

Different validity threats can arise in experiments. Concerning *internal threats*, it is a vulnerability the adoption of semi-real datasets. Despite the calibration being empirically obtained, a fine-tuning parametrization would lead to better recommendations in more complex scenarios. Applying other metaheuristics is attractive to contrast findings in terms of formulation suitability, solutions quality, processing cost, and ease of understanding and usability.

Regarding *external threats*, experiments considered 25 and 50 requirements. Replications and adaptations on a wider range of datasets with more requirements, risks, mitigation techniques and customers are desirable to achieve generalized findings, allowing to identify sources of biases. Indeed, the results should be applicable to situations where similar assumptions are held. Otherwise, *SR*$^2$ needs to be adapted accordingly, for instance in a context with interdependent requirements.

In relation to *construct threats*, *SR*$^2$ adopts concepts and information models which have been successfully applied in related work. The assumptions and rationale in this paper made sense for the type of experiments discussed, however, the formulation might change in the case that different abstractions for risks are perceived and adopted.

Referring to *conclusion threats*, as a mean to counter the stochastic nature of search techniques and ensure a fair comparison, NSGA-II and random search strategies were performed multiple independent runs for each experiment, overcoming randomness inherent in such strategies. In complement, a more valuable statistical analysis needs to be conducted, measuring statistical similarities and differences among MOP algorithms.

Thus, regarding future work, *SR*$^2$ ought to be evaluated in real, large-scale software projects, mitigating some validity threats, without introducing synthetical data. As another future but already started initiative, *SR*$^2$ is under evaluation using different MOP algorithms, such as SPEA2 and MOCell. Other possible branch for *SR*$^2$ could be to reshape risk probability and severity into fuzzy functions, instead of relying upon fuzzy-terms statically mapped to specific values, leading to better manipulation of inherent uncertainties related to risk priority and severity.

## REFERENCES

Alam, I., 2014. Role of software metrices in identifying the risk of project. *International Journal of Advancement in Engineering Technology, Management & Applied Science*, 1(1), 16-22.

Bagnall, A. J., Rayward-Smith, V. J. and Whittley, I. M., 2001. The next release problem. *Information and Software Technology*, 43(14), 883-890.

Baker, P., Harman, M., Steinhofel, K. and Skaliotis, A., 2006. Search based approaches to component selection and prioritization for the next release problem. In *22nd IEEE International Conference on Software Maintenance (ICSM 2006)*. Philadelphia: IEEE. 176-185.

Bannerman, P. L., 2008. Risk and risk management in software projects: a reassessment. *Journal of Systems and Software*, 81(12), 2118-2133.

Boehm, B., 1991. Software risk management: principles and practices. *IEEE Software*, 8(1), 32-41.

Brasil, M. *et al.*, 2011. A multiobjective optimization approach to the software release planning with undefined number of releases and interdependent requirements. In *International Conference on Enterprise Information Systems (ICEIS 2011)*. Beijing: Springer. 300-314.

Colares, F. *et al.*, 2009. A new approach to the software release planning. In *23rd Symposium on Software Engineering (SBES 2009)*. Fortaleza: IEEE. 207-215.

Dantas, A., Yeltsin, I., Araújo, A. A. and Souza, J., 2015. Interactive software release planning with preferences base. In *7th International Symposium on Search-Based Software Engineering (SSBSE 2015)*. Bergamo: Springer. 341-346.

Deb, K., 2001. *Multi-objective optimization using evolutionary algorithms*. 1st. ed. New York: John Wiley & Sons.

Deb, K., Pratap, A., Agarwal, S. and Meyarivan, T., 2002. A fast and elitist multiobjective genetic algorithm: NSGA-II. *Transactions on Evolutionary Computing*, 6(2), 182-197.

Durillo, J. *et al.*, 2011a. A study of the bi-objective next release problem. *Empirical Software Engineering*, 16(1), 29-60.

Durillo, J. and Nebro, A., 2011b. jMetal: a Java framework for multiobjective optimization. *Advances in Engineering Software*, 42(10), 760-771.

Harman, M., McMinn, P., Souza, J. T. and Yoo, S., 2012. Search based software engineering: techniques, taxonomy and tutorial. In *Empirical Software Engineering and Verification*. Berlin: Springer. 1-59.

Huhe, G. and Greer, D., 2003. Quantitative studies in software release planning under risk and resource constraints. In *International Symposium on Empirical Software Engineering (ISESE 2003)*. Rome: IEEE. 262-271.

Karim, M. and Ruhe, G., 2014. Bi-objective genetic search for release planning in support of themes. In *International Symposium on Search Based Software Engineering (SSBSE 2014)*. Fortaleza: Springer. 123-137.

Karlsson, J. and Ryan, K., 1997. A cost-value approach for prioritizing requirements. *IEEE Software*, 14(5), 67-74.

Li, L., Harman, M., Letier, E. and Zhang, Y., 2014. Robust next release problem: handling uncertainty during optimization. In *Annual Conference on Genetic and Evolutionary Computation (GECCO 2014)*. Vancouver: ACM. 1247-1254.

Pitangueira, A. M., Maciel, R. S. P. and Barros, M., 2015. Software requirements selection and prioritization using SBSE approaches: a systematic review and mapping of the literature. Journal of Systems and Software, 103(C), 267-280.

Saraiva, R., Araújo, A. A., Dantas, A. and Souza, J., 2016. A multiobjective approach based on interactive optimization for release planning. In *7th Brazilian Workshop on Search Based Software Engineering (WESB 2016)*. Maringá: SBC. 31-40.

Scacchi, W., 2001. Process models in software engineering. In Marciniak, J. J. (ed.), *Encyclopedia of Software Engineering*, 2nd ed. New York: John Wiley and Sons. 993-1005.

Verner, J., Sampson, J. and Cerpa, N., 2008. What factors lead to software project failure? In *2nd International Conference on Research Challenges in Information Science (RCIS 2008)*. Marrakech: IEEE. 71-80.

Yang, L., Jones, B. F. and Yang, S. H., 2006. Genetic algorithm based software integration with minimum software risk. *Information and Software Technology*, 48(3), 133-141.

Zhang, H., 2007. A redefinition of the project risk process: using vulnerability to open up the event-consequence link. *International Journal of Project Management*, 25(7), 694-701.

Zitzler, E and Thiele, L., 1999. Multi-objective evolutionary algorithms: a comparative case study and the strength pareto approach. *IEEE Transactions on Evolutionary Computation*, 3(4), 257-271.