MULTIOBJECTIVE SOFTWARE RELEASE PLANNING WITH DEPENDENT REQUIREMENTS AND UNDEFINED NUMBER OF RELEASES

Márcia Maria Albuquerque Brasil, Thiago Gomes Nepomuceno da Silva, Fabrício Gomes de Freitas, Jerffeson Teixeira de Souza and Mariela Inés Cortés
Optimization in Software Engineering Group (GOES.UECE), State University of Ceara (UECE)
Av. Paranjana 1700, Fortaleza, Ceara, 60740-903, Brazil
{marcia.abrasil, thi.nepo, fabriciogf.uece}@gmail.com, {jeff, mariela}@larces.uece.br

Keywords: Software release planning, Search-based software engineering, Requirements engineering, Multiobjective optimization, Genetic algorithms.

Abstract: Release Planning is an important and complex activity in software development. It involves several aspects related to which functionalities are going to be developed in each release of the system. Consistent planning must meet the customers' needs and comply with existing constraints. Optimization techniques have been successfully applied to solve problems in the Software Engineering field, including the Software Release Planning Problem. In this context, this work presents an approach based on multiobjective optimization for the problem when the number of releases is not known *a priori* or when the number of releases is a value expected by stakeholders. The strategy regards on the stakeholders' satisfaction, business value and risk management, as well as provides ways for handling requirements interdependencies. Experiments show the feasibility of the proposed approach.

1 INTRODUCTION

Software Release Planning consists of scheduling a set of requirements in a sequence of releases in order to meet customers' needs and to attend the related constraints. In a development process based on small deliveries, the software is not fully developed at once, by the work being focused on small and frequent releases, and with each release, a subset of functionality is delivered. In this way, customers receive software features sooner rather than waiting a long time to get the complete system. This development model has several positive aspects, such as: earlier feedback from the stakeholders, higher risk management, and allows incremental tests execution (Colares et al., 2009).

Deciding which requirements will be developed in each release is not an easy task and involves several aspects, sometimes conflicting. These aspects regard on balancing the customer satisfaction, business value, priority, involved risks, delivery time, available resources, and requirements interdependencies, among others.

An important and difficult aspect in Release Planning is to decide how many releases will be necessary to deliver the functionalities. Also, the clients and stakeholders may ask for a number of releases, then this issue has also to be considered.

This work addresses the Software Release Planning and presents an approach based on multiobjective optimization to assist project managers in an effective planning. The proposed method aim to determine the ideal number of releases required to develop the requirements considering important aspects of real projects.

Search techniques have been successfully applied for solving complex Software Engineering problems. The model proposed in this paper belongs to this recent and promising research field called Search-Based Software Engineering – SBSE (Harman and Jones, 2001), which consists in solving Software Engineering problems mathematically modelled through use of optimization techniques. The SBSE approach is effective in providing better solutions to these problems, in comparison to human generated solutions (Souza et al., 2010).

This paper presents the following contributions:

- Introduces a multiobjective formulation for the Software Release Planning when the number of releases is not determined in advance or it is an expectation among stakeholders. The method considers important aspects of the problem, such as: customer satisfaction, business value, risk management, available resources. The proposed approach deals with requirements interdependencies including technical precedence, coupling and business precedence;
- Uses multiobjective metaheuristics based on genetic algorithms to solve problem instances;
- Presents results of experiments conducted in order to demonstrate the feasibility and efficiency of the proposed formulation.

This paper is organized as follows. Section 2 discusses related works in the Requirements Prioritization and in the Software Release Planning. Section 3 presents important aspects and definitions considered in the problem and describes the requirements interdependencies addressed in this work. In Section 4, the proposed approach is explained and formally defined. Section 5 is devoted to explain some concepts of multiobjective optimization and describes the algorithms used in the experiments. Section 6 regards the experiments and provides a discussion about the results. Finally, Section 7 concludes and outlines future research.

2 RELATED WORK

Karlsson and Ryan (1997) develop an approach based on cost/value to prioritize requirements using the Analytic Hierarchy Process (Saaty, 1980) method to compare requirements pair wise based on their value and implementation cost. Jung (1998) presents a variant of the 0-1 knapsack problem to reduce the complexity in the cost/value approach.

The requirements selection was initially addressed in (Bagnall, Rayward-Smith and Whittley, 2001), the "The Next Release Problem" (NRP), which consists on the selection of which customers will be met in the next release. The approach prioritizes the most important customers and complies with the available resources and requirements precedence. Several techniques were employed, including Integer Programming, GRASP, Hill Climbing and Simulated Annealing. In this mono-objective formulation, the release planning is defined only for the next release and does not consider requirement value for customers.

Later, Greer and Ruhe (2004) present an iterative approach based on genetic algorithms to the Software Release Planning. The method is called EVOLVE and provides decision support in a changing environment. The objective function is a linear combination of two functions, aimed to maximize the total benefit and minimize the total penalties. The number of releases is not decided *a priori* and re-planning future releases is allowed.

Ruhe and Saliu (2005) propose a hybrid method that combines computational algorithms with human knowledge and experience. The objective function determines the value for the weighted average satisfaction according stakeholders priorities for all features. They solved the problem using Integer Linear Programming and the approach was implemented as part of a intelligent decision-support tool. Saliu and Ruhe (2005) present some technical and nontechnical factors affecting release planning, and evaluate methods based on these aspects. They also propose a framework considering the impact of existing systems characteristics to make decisions about release planning.

A multiobjective formulation for NRP was presented by (Zhang, Harman and Mansouri, 2007). Customer satisfaction and project cost were the objectives to be optimized, when selecting the optimal requirements set. Four different multiobjective optimization techniques were used, including NSGA-II. However, their formulation does not include any requirement interdependence, which is uncommon in the context of real projects, since one requirement may depend on another in different ways (Carlshamre et al., 2001).

(Saliu and Ruhe, 2007) present a technique for detecting coupling between features from the implementation perspective. The work focuses in the evaluation of the release plans from business perspective and based on relationships between the components that would realize the features.

A multiobjective approach to the Software Release Planning is proposed in (Colares et al., 2009). The formulation tries to be complete and aims to maximize customer satisfaction and minimize project risks, by selecting the requirements to be developed in a fixed number of releases. Customer satisfaction is achieved by implementing earlier the highest priority requirements and the project risks are minimized by implementing the requirements with higher risk first. NSGA-II is applied to solve the problem and the human-competitiveness of the approach is also studied.

An overview on requirements optimization is available in (Zhang, Finkelstein and Harman, 2008).

3 PROBLEM DEFINITION

3.1 Important Aspects Considered

This subsection describes important aspects related to the proposed approach. The requirements interdependencies are explained in a separated subsection. Numeric scales used are just a way to evaluate values for risk, importance, priority, time and cost to enable the mathematical modelling of the problem. Other scales can be used in other contexts.

3.1.1 Requirements

Let be $R = \{r_i | \, i = 1, 2, \ldots, N\}$ the set of functionalities to be developed and assigned to releases. System requirements include features, functions and attributes in the software system (Karlsson and Ryan, 1997). The implementation of each requirement r_i demands a certain amount of cost and time denoted by $cost_i$ and $time_i$, respectively. Each requirement r_i has also an associated risk denoted by $risk_i$, ranging on a scale of 1 (lower risk) to 5 (higher risk).

3.1.2 Stakeholders

Let be $C = \{c_m | \, m = 1, 2, \ldots, M\}$ the set of stakeholders involved in the system development. Stakeholders play an important role in release planning, by influencing or are affecting the process. They may include customers and users, a software engineer, developers and so on. For each stakeholder c_m it is assigned a weight based on their relative importance to the company. Thus, w_m defines the importance of a stakeholder to the software organization and is quantified on a scale from 1 (lower importance) to 10 (higher importance).

3.1.3 Releases

Let be $S = \{s_k | \, k = 1, 2, \ldots\}$ the set of releases. The total amount of releases is not initially defined. Instead it will be determined by the approach. For each release s_k, it is given an interval for budget and duration, denoted by $budgetReleaseMin_k$and $budgetReleaseMax_k$, and by $timeReleaseMin_k$ and $timeReleaseMax_k$, respectively.

3.1.4 Project

The whole project consists of all the releases that should be planned. Thus, the project has a maximum schedule ($timeProject$) and a total budget ($budgetProject$) that should not be exceeded. These values are used to select, during the prioritization, the requirements to be implemented.

3.1.5 Requirements *versus* Stakeholders

Different stakeholders may have different interests in the implementation of each requirement. Just as in (Greer and Ruhe, 2004) and in (Ruhe and Saliu, 2005), the concepts of priority, in terms of urgency, and value, in terms of business value-added, are used in this work. These concepts are analyzed from the stakeholders' perspective. Thus, $value(m, i)$ quantifies the perceived importance that a stakeholder c_m associates to a requirement r_i by assigning a value ranging from 0 (no importance) to 10 (highest importance), and $priority(m, i)$ denotes the urgency that a stakeholder c_m has for the implementation of requirement r_i, ranging from 0 (no urgency) to 10 (highest urgency).

3.2 Requirements Interdependencies

According to a study in (Carlshamre et al., 2001), 75% of interdependencies come from approximately 20% of the requirements. The set of requirements interdependencies addressed in this work are:

- Coupling (T1) – It is DESIRABLE that a requirement r_i be implemented together with a requirement r_j, i.e. in the same release.
- Business Precedence (T2) – It is DESIRABLE that requirement r_i be implemented before r_j. In this case, r_i can be implemented in a previous or the same release of requirement r_j.
- Technical Precedence (T3) – A requirement r_i MUST BE implemented before a r_j. The requirement r_i must be implemented in a previous or the same release of requirement r_j.

For example, when two requirements have a Technical Precedence relationship to each other, the interdependence between r_i and r_j is denoted by $Dr_i, r_j = T3$. The same applies to the other forms.

T1 and T2 relationships are objectives to be optimized in the proposed approach. Reducing or minimizing the coupling between releases means implementing requirements with similar characteristics in the same release. In fact, the detection of coupling and implementation of requirements with similar characteristics bring benefits, as reuse and resources saving (time and effort) (Saliu and Ruhe, 2007). T2 expresses relationships from the business viewpoint according to stakeholders' perspectives.

4 MATHEMATICAL DEFINITION

The approach proposed in this paper for release planning is divided into two methods. The first one is used when the number of releases is not initially defined and the approach will try to determine this number. In the second one, the number of releases is also undetermined *a priori*, but exist a consensual value expected by the stakeholders and the approach will try to reach this value. Each of these methods is performed in two phases, where the first phase is common for both.

4.1 Release Planning with Undefined Number of Releases

4.1.1 Phase 1: Requirements Prioritization

In this phase, the requirements will be prioritized according to the goals of value, priority and risk, while respecting technical precedence and available resources (overall time and budget). Due to constraints of budget and time of the project, it is possible that not all requirements from the initial set of requirements are selected.

Therefore, the first phase for software release planning can be mathematically formulated with the following objective and constraint functions:

$$Max\ f_{VALUE}(y) = \sum_{i=1}^{N} score_i . y_i \tag{1}$$

$$Max\ f_{PRIORITY}(x^{Prio}, y) = \sum_{i=1}^{N} prior_i . (N - x^{Prio}{}_i) . y_i \tag{2}$$

$$Min\ f_{RISK}(x^{Prio}) = \sum_{i=1}^{N} risk_i . x^{Prio}{}_i \tag{3}$$

Subject to:

$$\sum_{i=1}^{N} cost_i . y_i \leq budgetProject \tag{4}$$

$$\sum_{i=1}^{N} time_i . y_i \leq timeProject \tag{5}$$

$$x^{Prio}{}_i < x^{Prio}{}_j, if\ Dr_i, r_j = T3\ (Technical\ Precedence) \tag{6}$$

The variable $x^{Prio}{}_i$ indicates the position of the requirement r_i in the established prioritization. It is a value in $\{0,\ 1, 2,\ \ldots N\}$ for $i = 1, 2,\ \ldots N$. The variable y_i indicates if the requirement r_i will be implemented ($y_i = 1$) or not ($y_i = 0$), for $i = 1, 2, \ldots N$. If $x^{Prio}{}_i > 0, y_i = 1; y_i = 0$, otherwise.

Function 1 – This objective function expresses the stakeholders' satisfaction by implementing the most important requirements, where $score_i = \sum_{m=1}^{M} w_m . value(m, i)$ means the weighted business-value added by the development of requirement r_i.

Function 2 – This objective function expresses, in a weighted way, the customer satisfaction for the early implementation of the highest priority requirements ($prior_i = \sum_{m=1}^{M} w_m . priority(m, i)$).

Function 3 – This objective function expresses the project risk management as a whole. Requirements with a high risk associated are more likely to give problems in development (Sommerville and Sawyer, 1997). Thus, in the same way that (Colares et al., 2009), requirements at higher risk should be implemented earlier.

The constraints of this phase are expressed in 4, 5 and 6. Thus, (4) is the restriction that limits the cost of implementing to the overall project budget. And (5) is the restriction that limits the time necessary of implementation to the overall project duration. The constraint (6) expresses technical precedence between requirements. If a requirement r_i technically precedes a requirement r_j, then r_i should be implemented before r_j ($x^{Prio}{}_i < x^{Prio}{}_j$).

4.1.2 Phase 2: Scheduling in Releases

The phase 2 will allocate the requirements (selected and prioritized in the first phase) in releases. The approach tries to put in the same release the requirements with a coupling interdependency. It also tries to maintain the established prioritization and respect the existing business precedence. The mathematical formulation follows:

$$Min\ f_{COUPLING}(x^{Rel}) = C_{Rel}(x^{Rel}) \tag{7}$$

$$Min\ f_{PRIORITIZATION\_ORDERING}\ (x^{Prio}, x^{Rel}) = PO(x^{Prio}, x^{Rel}) \tag{8}$$

$$Min\ f_{BUSINESS\_PRECEDENCE}(x^{Rel}) = \sum_{i=1}^{N} \sum_{j=1}^{N} BP(x^{Rel}{}_i, x^{Rel}{}_j) \tag{9}$$

Subject to:

$$budgetReleaseMin_k \leq \sum_{i=1}^{N} cost_i . v_{i,k} \leq budgetReleaseMax_k, \forall k \in \{1, 2, \ldots\} \tag{10}$$

$$timeReleaseMin_k \leq \sum_{i=1}^{N} time_i . v_{i,k} \leq timeReleaseMax_k, \forall k \in \{1, 2, \ldots\} \tag{11}$$

$$x^{Rel}{}_i \leq x^{Rel}{}_j, if\ Dr_i, r_j = T3\ (Technical\ Precedence) \quad (12)$$

The variable $x^{Prio}{}_i$ indicates the position of r_i in the prioritization. The variable $x^{Rel}{}_i$ denotes the release for implementation of the requirement r_i, and is a value in $\{1, 2, \ldots\}$ for $i = 1, 2, \ldots N$. The variable $v_{i,k}$ indicates if the requirement r_i is implemented in the release s_k ($v_{i,k} = 1$) or not ($v_{i,k} = 0$).

Function 1 – This objective function aims to minimize the coupling between releases, according to the strategy presented in (Carlshamre et al., 2001).

$$C_{Rel}(x^{Rel}) = \left[\frac{1}{2}.\sum_{i=1}^{N}\sum_{j=1}^{N} C(x^{Rel}{}_i, x^{Rel}{}_j)\right]/I \quad (13)$$

$$C(x^{Rel}{}_i, x^{Rel}{}_j) = \begin{cases} 1, & if\ Dr_i, r_j = T1\ and\ x^{Rel}{}_i \neq x^{Rel}{}_j \\ 0, & otherwise \end{cases} \quad (14)$$

$$I = \frac{1}{2}.\sum_{i=1}^{N}\sum_{j=1}^{N} C_{Req}(r_i, r_j) \quad (15)$$

$$C_{Req}(r_i, r_j) = \begin{cases} 1, & if\ Dr_i, r_j = T1 \\ 0, & if\ Dr_i, r_j \neq T1 \end{cases} \quad (16)$$

Function 2 – This function tries to maintain the prioritization ordering obtained in the phase 1 and counts negatively when this sequence is broken:

$$PO(x^{Prio}, x^{Rel}) = \sum_{i=2}^{N}\sum_{j=1}^{i-1} Violation(z[i], z[j]) \quad (17)$$

$$Violation(z[i], z[j]) = \begin{cases} 1, if\ x^{Rel}[z[i]] < x^{Rel}[z[j]] \\ 0, \quad otherwise \end{cases} \quad (18)$$

$$\begin{aligned} z[i] &= A, for\ each\ x^{Prio}[A] = i \\ z[j] &= A, for\ each\ x^{Prio}[A] = j \end{aligned} \quad (19)$$

The vector z is an auxiliary vector to sort the requirements according to the prioritization and to compare their elements with the release implementation of each requirement to verify if the ordering obtained in the phase 1 was not followed.

Function 3 – This function tries to minimize the amount of Business Precedence that was not fulfilled. This situation occurs when a requirement r_i is prerequisite, from the business perspective, for a requirement r_j but is allocated in a later release.

$$BP(x^{Rel}{}_i, x^{Rel}{}_j) = \begin{cases} 1, if\ Dr_i, r_j = T2\ and\ x^{Rel}{}_i > x^{Rel}{}_j \\ 0, \quad otherwise \end{cases} \quad (20)$$

Equations 10, 11 and 12 are the constraints. Restriction (10) limits the implementation cost in a release to the interval of budget available for this release. Restriction (11) limits the implementation time in a release to the interval of schedule available for this release. Restriction (12) is the same as in phase 1. But in this case if r_i technically precedes r_j, then r_i should be implemented in a release prior to the release of r_j, or both should be implemented in the same release ($x^{Rel}{}_i \leq x^{Rel}{}_j$).

4.2 Release Planning with Expected Number of Releases

This approach is very similar to the first one. The overall formulation remains the same (it is executed in two phases and has the same objectives and constraints) and one more objective function is added, as below:

$$Min\ f_{TARGET\_DISTANCE}(K) = |K - TR| \quad (21)$$

Function 7 – This function tries to reach the number of releases wanted by the stakeholders (the target release). K is the number of releases obtained by the approach and $target_m$ is the number of releases expected by the stakeholder c_m. The expected number of releases is obtained in a weighted form, according to:

$$TR = \sum_{m=1}^{M}(w_m.target_m)/\sum_{m=1}^{M} w_m \quad (22)$$

5 MULTIOBJECTIVE OPTIMIZATION

Since the problem addressed in this paper is modeled as a multiobjective optimization problem, this section presents some concepts related to multiobjective optimization and describes the algorithms, NSGA-II (Deb et al., 2002) and MOCell (Nebro et al., 2009), used in the experiments.

5.1 Pareto Front

In multiobjective optimization problems, two or more functions must be optimized, the solutions are partially ordered and the search is done by a set of solutions. Often, many real world optimization problems have conflicting goals and involve the minimization and/or maximization of more than one function. When trying to optimize multiple objectives simultaneously, the search space becomes partially ordered and is based on the concept of dominance. The search is not restricted to find a single solution. Instead, it returns a set of solutions called non-dominated solutions (solutions that are not dominated by any other solution). A solution S_1 dominates a solution S_2 if S_1 is better or equal to S_2 in all objectives and strictly better in at least one of them. Each solution in the set of non-dominated solutions is said to be Pareto optimal. The collective representation of all these solutions is called Pareto front.

5.2 NSGA-II

NSGA-II – Non-dominated Sorting Genetic Algorithm II (Deb et al., 2002) is a metaheuristic based on genetic algorithms for multiobjective optimization, which implements the concepts of dominance and elitism, classifying the population into different quality categories (fronts) according to dominance degree. It is implemented in two phases, using the algorithms: Non-dominated Sorting Algorithm (performs a search for solutions near the Pareto front) and Crowding Distance (performs a search for solutions with a good distribution in space). The execution begins with an entire population not yet classified. Next, each individual is assigned with a degree of dominance over all other individuals from this population, through the comparison between individuals. Following, the individuals are classified into fronts according to their dominance value. Thus, in the first front will be ranked the best individuals and in the last front, the worst. This process continues until all individuals are classified into a front. In the next phase, the individuals are classified according to the diversity operator (crowding distance). This operator orders each individual according their distance to neighboring points of the same front, related to each goal. The greater the distance, greater the probability of being selected. This mechanism enables a better spread of solutions.

5.3 MOCell

MOCell (Nebro et al., 2009) is an adaptation of a cellular model of genetic algorithm (*cGA - cellular genetic algorithm*) canonical for multiobjective optimization. The algorithm uses an external file to store non-dominated solutions found during the search and applies a feedback mechanism in which the solutions in this file replace, randomly, existing individuals in population after each iteration. To manage the insertion of solutions in the *Pareto front*, for a diverse set, a density estimator (based on *crowding distance* method) is used. This mechanism is also used to remove solutions from the archive when it becomes full (the external file has a maximum size). The algorithm starts by creating an empty *Pareto front*. Individuals are organized into a two-dimensional grid and genetic operators are successively applied to them until a stop condition is reached. For each individual, the algorithm selects two parents from their neighborhood, makes a recombination between them in order to obtain a descendant, executes a mutation and evaluates the resulting individual. This individual is inserted both in helping population (if not dominated by the current individual) as in the *Pareto front*. After each iteration, the old population is replaced by an assistant and a feedback procedure is triggered to replace a fixed number of individuals in the population, randomly selected, for solutions from the file. It is an elitist algorithm useful for obtaining competitive results in terms of both convergence and diversity of solutions along the *Pareto front*.

5.4 Metrics for Comparison

At least two performance metrics must be used when comparing algorithms for multiobjective optimization: one to evaluate the spread of solutions and another to assess the progress toward the Pareto-optimal front (Deb, 2009). In this work, the metrics *Spread* and *Hypervolume* were used for comparing the performance of algorithms.

Spread (Deb, 2009) is used for evaluating diversity among non-dominated solutions. An ideal distribution has zero value for this metric. The first condition for this is there exists the true extreme Pareto-optimal solutions in the obtained set of non-dominated solutions. And the second is that the intermediate solutions are uniformly distributed. The closer to zero, better the distribution. An algorithm that achieves a smaller value for Spread can get a better diverse set of non-dominated solutions.

Hypervolume (Deb, 2009) provides a qualitative measure of convergence and diversity. In problems of minimization (all objectives must be minimized), it calculates the volume covered by members of the set of non-dominated solutions in the objective space. An algorithm finding a large value of this metric is desirable.

5.5 Present Results

In a multiobjective problem with two objectives, a two-dimensional objective space plot is used to show the performance achieved by the metaheuristics by illustrating the obtained non-dominated solutions. In the multiobjective problem modeled in this work, there are more than two objectives, for each approach, for each phase.

In a situation like that, when number of objectives is greater than two, a representation in a two-dimensional space is difficult and the obtained non-dominated solutions can be represented through several illustration techniques (Deb, 2009). In this work, the scatter-plot matrix method (Deb, 2009) was applied.

6 EXPERIMENTS, RESULTS AND ANALYSIS

This section describes the experiments and presents the results of a preliminary evaluation conducted to demonstrate the feasibility of the proposed approach.

6.1 Experiments

6.1.1 Description of the Instances

Three different instances of problems were randomly generated and used to analyze the proposed approach in different contexts. The datasets are used as a mean of simulating a generic context. Therefore, without loss of generality, the instances represent a practical application scenario, and its use with the aiming of analysis of the proposed approach is valid.

The values for $risk_i$,$value(m,i)$, $priority(m,i)$ were generated according to the scales defined. The values for $cost_i$, $time_i$ and w_m were randomly generated using scales from 10 to 20, 10 to 20 and 1 to 10, respectively. The values for $timeProject$ and $budgetProject$ were considered as 70% of the necessary resources to implement all requirements. The other values (range of resources necessary for each release) were also randomly obtained. Matrices of interdependencies were randomly generated according to the kinds of relationships defined in subsection 3.2, with 10% of interdependencies. Table 1 shows the attributes of each instance.

Table 1: Features of the Instances.

Instance	Requirements	Stakeholders	Releases (Consensual value)
Inst. A	30	3	5
Inst. B	50	5	10
Inst. C	80	8	6

6.1.2 Parameter Settings

To solve the problem formulated, the metaheuristics NSGA-II and MOCell were applied. The parameters used for each method were set from execution of preliminary tests and are showed in table 2 below.

Table 2: Parameter Settings.

Metaheuristic	NSGA-II	MOCell
Population Size	250	256
Stopping Criteria (evaluations)	32,000	32,768
Crossover Rate	0.9	0.9
Mutation Rate (N – requirements)	$1/N$	$1/N$

6.1.3 Framework *jMetal*

The implementation for the proposed approach was performed using a known framework, called *jMetal* (Durillo et al., 2006), which provides metaheuristics for multiobjective optimization, including NSGA-II and MOCell.

6.2 Results

In this work, the average and the standard deviation of two executions of the algorithms in each instance and in each approach were calculated. In addition to the *Spread* and *Hypervolume* metrics, the execution time was also computed (in milliseconds).

The results obtained in each instance for each approach are presented below. Because of space limitations, only some important graphics are presented.

6.2.1 Results for Release Planning with Undefined Number of Releases

The tables 3, 4 and 5 show the results for performance of the two algorithms for each instance using the metrics.

Table 3: *Spread*.

Instance Name	NSGA-II	MOCell
Inst. A	1.875096	0.261888
Inst. B	1.027938	0.275178
Inst. C	0.488216	0.337055

Table 4: *Hypervolume*.

Instance Name	NSGA-II	MOCell
Inst. A	0.320515	0.288779
Inst. B	0.335314	0.291431
Inst. C	0.407802	0.400079

Table 5: Execution time (in milliseconds).

Instance Name	NSGA-II	MOCell
Inst. A	6247.755725	1415.790476
Inst. B	4463.549180	3264.715517
Inst. C	8721.25	6798.402515

The graphs below show the results for the execution of the two algorithms in the instances. The Figure 1 presents the results for the smallest instance, where we can see that the two algorithms have found solution in different areas of the search space. This has also happened in the two others instances, B and C, which have their results for this approach respectively shown in Figure 2 and Figure 3, as presented below.

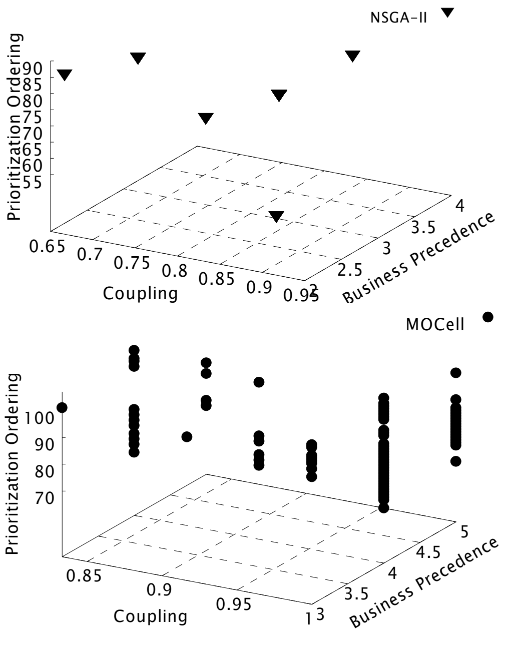

Figure 1: Results for instance A.

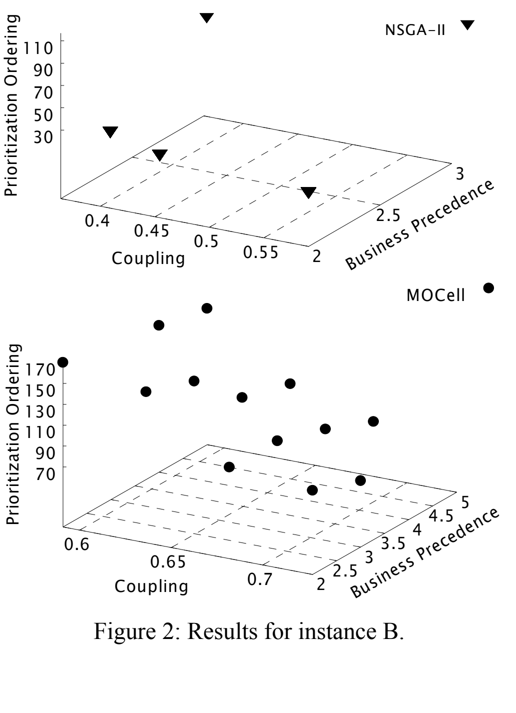

Figure 2: Results for instance B.

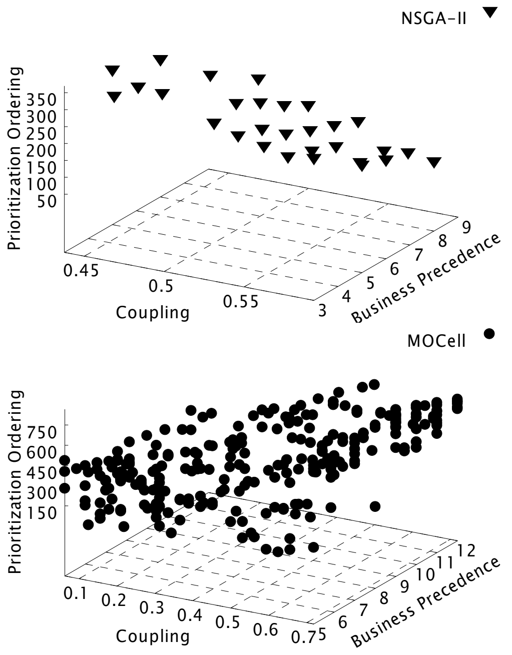

Figure 3: Results for instance C.

6.2.2 Results for Release Planning with Expected Number of Releases

The tables 6, 7 and 8 show the results for performance of the three algorithms for each instance using the metrics.

Table 6: *Spread.*

Instance Name	NSGA-II	MOCell
Inst. A	1.876102	0.243523
Inst. B	1.695456	0.441614
Inst. C	1.612609	0.432743

Table 7: *Hypervolume.*

Instance Name	NSGA-II	MOCell
Inst. A	0.224599	0.248997
Inst. B	0.339315	0.535095
Inst. C	0.302255	0.381188

Table 8: Execution time (in milliseconds).

Instance Name	NSGA-II	MOCell
Inst. A	2893.175438	1473.625
Inst. B	4539.538461	3933.222543
Inst. C	6524.685393	6729.600798

The figures 4 and 5 show the results for the execution of both algorithms for instances B and C.

6.3 Analysis

The figures 1, 2 and 3 show the solutions generated by the NSGA-II and MOCell metaheuristics in the instances A, B and C, respectively, for the first approach. As a result from the experiments, we can indicate that the use of both techniques is desired for this problem since the algorithms have found solutions in different areas. The plots also indicate that it is possible to choose a solution taking in account, for instance, the business-precedence wanted for the decision maker. By each possible value in this objective, there is a set of solutions optimized to the others objectives.

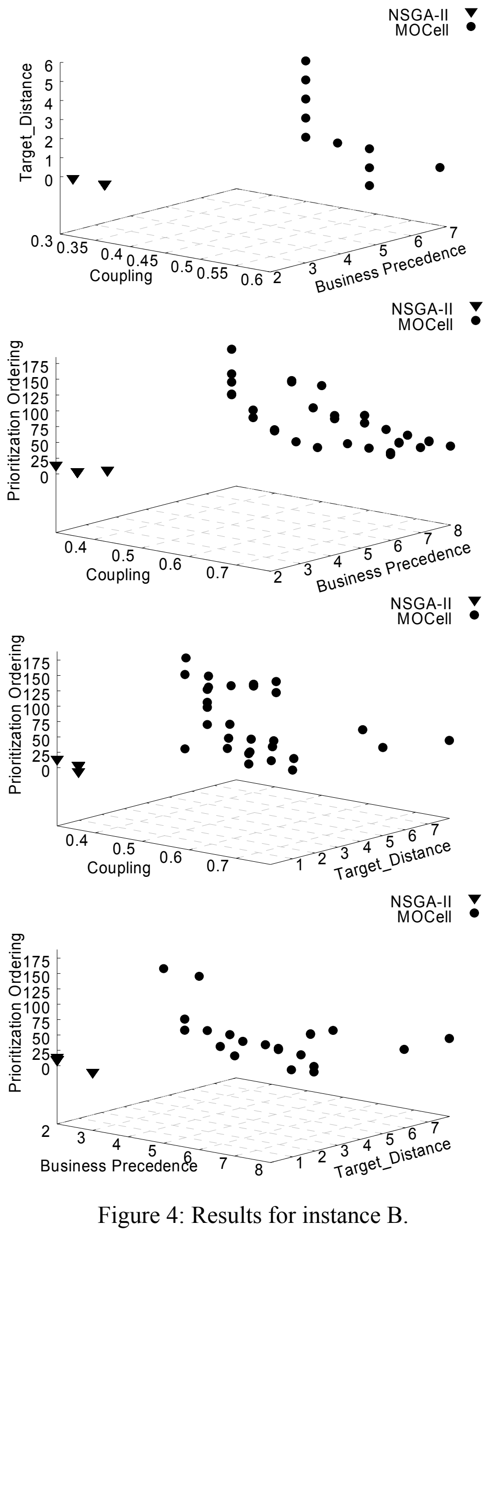

Figure 4: Results for instance B.

Additionally, we show next, in figures 4 and 5 the solutions to the second approach, which deals with expected number of releases. The formulation to this approach has four objectives. In order to be able to show the results in an effective way, we have decomposed the four-dimensional solution space in four tridimensional graphics. These graphics allow a better visualization of the solutions, regarding different combinations of the objectives. In addition, information in the graphs is combined referent to the original solutions.

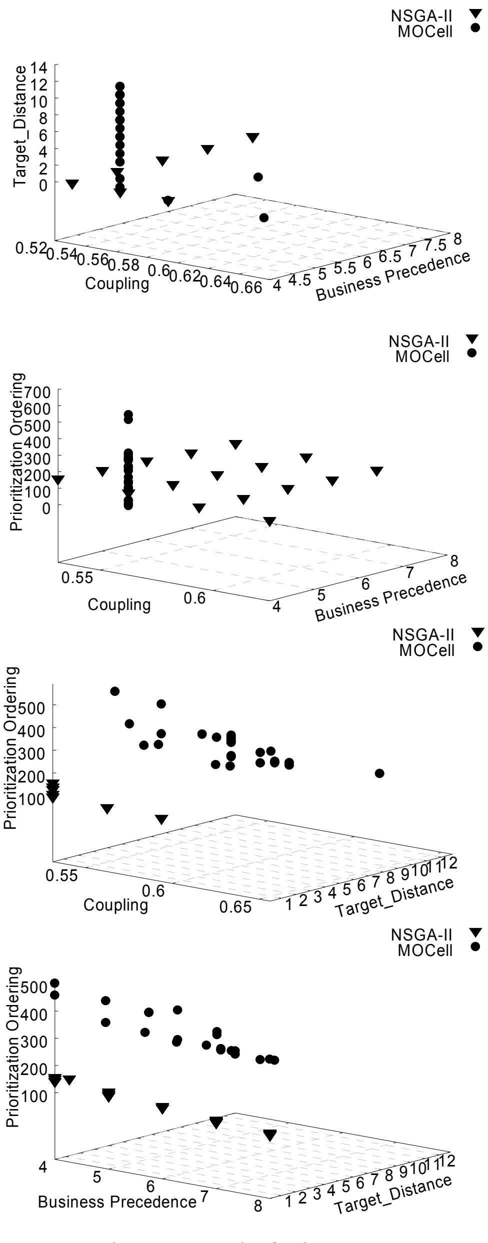

Figure 5: Results for instance C.

For this approach, few solutions were found, which demonstrates the complexity of the problem and thus an indication that your resolution manually would be inappropriate and inefficient. The results from the metrics in the metaheuristics indicate that both MOCell and NSGA-II have good results. This is confirmed by the better spread value in all instances for the MOCell, and the better value of hypervolume for NSGA-II. The execution time of MOCell has been show generally better.

7 CONCLUSIONS

According to Greer and Ruhe (2004), three things must be taken into consideration when planning releases: the technical precedence intrinsic to requirements, the conflicting priorities established by the most important stakeholders and the balance between the necessary and available resources.

In this study, the Software Release Planning problem was addressed as completely as possible, considering different aspects in a way closer to real practice environment. Thus, the proposed approach has a broader applicability.

The problem was solved using elitist multiobjective evolutionary algorithms on artificial data. Since search techniques have been successfully applied to solve problems in Software Engineering, the alternative release plans generated provide better support for decision making.

One negative aspect of this work was the amount and size of the instances used. Although the approach has proved feasible in the context used, more experiments are necessary in order to generalize it. Thus, future work includes further analysis of these preliminary results and definition and evaluation of other instances and studies using real-world data sets.

REFERENCES

Bagnall, A. J., Rayward-Smith, V. J., Whittley, I. M., 2001. The Next Release Problem. *Information and Software Technology*, 43(14):883–890.

Carlshamre, P., Sandahl , K., Lindvall, M., Regnell, B., Dag, J. N., 2001. An Industrial Survey of Requirements Interdependencies in Software Product Release Planning. In *Proceedings of the Fifth IEEE International Symposium on Requirements Engineering*, pages 84-91, Toronto, Canada. IEEE Computer Society.

Colares, F., Souza, J., Carmo, R., Padua, C., Mateus, G. R., 2009. A New Approach to the Software Release Planning. In *Proceedings of the XXIII Brazilian Symposium on Software Engineering, 2009 (SBES '09)*, pages 207-215, Fortaleza, Ceará, Brazil. IEEE Computer Society.

Deb, K., 2009. *Multi-Objective Optimization Using Evolutionary Algorithms*. Wiley.

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

Durillo, J. J., Nebro, A. J., Luna, F., Dorronsoro, B., Alba, E., 2006. *jMetal: a Java Framework for Developing Multi-Objective Optimization Metaheuristics.* Technical Report: ITI 2006-10, University of Málaga.

Greer, D., Ruhe , G., 2004. Software Release Planning: An Evolutionary and Iterative Approach. *Information & Software Technology*, 46(4):243–253.

Harman, M., Jones, B. F., 2001. Search-Based Software Engineering. *Information & Software Technology*, 43(14):833-839.

Jung, H.-W., 1998. Optimizing Value and Cost in Requirements Analysis. *IEEE Software*, 15(4): 74-78.

Karlsson, J., Ryan, K., 1997. A Cost-Value Approach for Prioritizing Requirements. *IEEE Software*, 14(5): 67-74.

Nebro, A. J., Durillo, J. J., Luna, F., Dorronsoro, B., Alba, E., 2009. MOCell: A Cellular Genetic Algorithm for Multiobjective Optimization. *International Journal of Intelligent Systems*, 24:726-746.

Ruhe, G., Saliu, M. O., 2005. The Art and Science of Software Release Planning. *IEEE Software*, 22(6): 47–53.

Saaty, T. L., 1980. *The Analytic Hierarchy Process*. McGraw-Hill.

Saliu, O., Ruhe, G., 2005. Supporting Software Release Planning Decisions for Evolving Systems. In *Proceedings of 29th Annual IEEE/NASA on Software Engineering Workshop (SEW '05)*, pages 14-26. IEEE Computer Society.

Saliu, M. O., Ruhe, G., 2007. Bi-Objective Release Planning for Evolving Software Systems. In *Proceedings of the 6th joint meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering*, pages 105–114, Dubrovnik, Croatia. ACM.

Sommerville, I., Sawyer, P., 1997. *Requirements Engineering: A Good Practice Guide.* John Wiley & Sons.

Souza, J. T., Maia, C. L., Freitas, F. G., Coutinho, D. P., 2010. The Human Competitiveness of Search Based Software Engineering. In *Proceedings of the 2nd International Symposium on Search Based Software*

Engineering (SSBSE '10), pages 143-152, Benevento, Italy. IEEE.

Zhang, Y., Finkelstein, A., Harman, M., 2008. Search Based Requirements Optimisation: Existing Work and Challenges. Requirements Engineering: Foundation for Software Quality. Lecture Notes in Computer Science. Springer Berlin / Heidelberg. Pages 88-94. Volume 5025.

Zhang Y., Harman, M., Mansouri, S. A., 2007, The Multi-Objective Next Release Problem. In *Proceedings of the 9th annual Conference on Genetic and Evolutionary Computation (GECCO '07)*, pages 1129–1137, London, UK. ACM.