Interactive Software Release Planning with Preferences Base

Altino Dantas(✉), Italo Yeltsin, Allysson Allex Araújo, and Jerffeson Souza

Optimization in Software Engineering Group, State University of Ceará,
Doutor Silas Munguba Avenue 1700, Fortaleza 60714-903, Brazil
{altino.dantas,allysson.araujo,jerffeson.souza}@uece.br,
italo.medeiros@aluno.uece.br
http://goes.uece.br

Abstract. The release planning is a complex task in the software development process and involves many aspects related to the decision about which requirements should be allocated in each system release. Several search based techniques have been proposed to tackle this problem, but in most cases the human expertise and preferences are not effectively considered. In this context, this work presents an approach in which the search is guided according to a *Preferences Base* supplied by the user. Preliminary empirical results showed the approach is able to find solutions which satisfy the most important user preferences.

Keywords: Release planning · Interactive Genetic Algorithm · SBSE

1 Introduction

The decision about which requirements should be allocated in a set of releases is a complex task in any incremental software development process. Thus, release planning is known to be a cognitively and computationally difficult problem [1]. This problem involves many aspects, such as the customers needs and specific constraints [2].

The current SBSE approaches to the software release planning fail to effectively consider the users preferences. Therefore, the users can have issues accepting such results, given that their expertise was not properly captured in the decision process. On the other hand, when human expertise might be considered, Interactive Optimization can be applied. The main idea of this approach is to incisively incorporate the decision maker in the optimization process, allowing a fusion of his preferences and the objective aspects related to the problem [3].

Given this context, the Interactive Genetic Algorithm (IGA) arises. This algorithm is derived from the Interactive Evolutionary Computation (IEC) and is characterized by the use of human evaluations in the computational search through bioinspired evolutionary strategies [3]. However, repeated user evaluations can cause a well-known critical problem in IEC, the human fatigue [4]. This problem may result in a direct quality reduction of user evaluations, given the cognitive exhaustion.

M. Barros and Y. Labiche (Eds.): SSBSE 2015, LNCS 9275, pp. 341–346, 2015.
DOI: 10.1007/978-3-319-22183-0_32

Regarding to the application of search based techniques to release planning, in [5] was proposed a method called EVOLVE based on GAs to decision support, which was extended in [1] considering diversification as a means to approach the uncertainties. Moreover, in [6] was proposed an approach aimed at maximizing the client satisfaction and minimizing the risks of the project. Recently, Araújo and Paixão [7] propose an interactive approach with machine learning to NRP.

This paper proposes an interactive approach to software release planning which employs an IGA guided through a *Preferences Base* supplied by the user.

2 Proposed Approach

The proposed interactive approach is comprised of three components (Fig. 1).

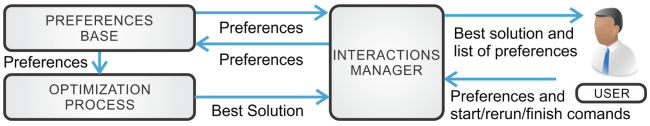

Fig. 1. Proposed approach components and their relations.

The *Interactions Manager* supports the user interactions, enabling the manipulation of the preferences, solutions visualization and control (start and finish) of the search process. The user preferences are stored in the *Preferences Base*. The *Optimization Process* is responsible to search solutions considering the *Preferences Base*.

Initially, through the *Interactions Manager*, the user defines his preferences, which are stored in the *Preferences Base*, and starts the *Optimization Process*. The best solution is shown after each execution of the search algorithm and the user can manipulate the preferences, rerun or stop the search process.

2.1 Release Planning Model

Consider a set of requirements $R = \{r_1, r_2, r_3, ..., r_N\}$ available to be selected for a set of releases $K = \{k_1, k_2, k_3, ..., k_P\}$, where N and P are the number of requirements and releases, respectively. Each requirement r_i has a implementation cost and risk defined by $cost_i$ and $risk_i$, respectively. Each release k_q has a budget constraint s_q. Thus, the requirements with highest risk should be allocated earlier and the sum of the costs of all requirements r_i allocated in k_q cannot exceed the s_q.

Consider $C = \{c_1, c_2, c_3, ..., c_M\}$ as the set of clients, where M is the number of clients and each client c_j has a degree of importance for the company that is reflected by a weight factor w_j. A requirement r_i might have a different value for each client defined by $importance(c_j, r_i)$ which represents how important the requirement r_i is to the client c_j. Finally, the solution representation is a vector $S = \{x_1, x_2, x_3, ..., x_N\}$ where $x_i \in \{0, 1, 2, ..., P\}$, where $x_i = 0$ implies that requirement r_i is not allocated, otherwise it is allocated in release k_q for $q = x_i$.

2.2 Model of User Preferences for Release Planning

The *Preferences Base* contains a set of preference assertions and their respective importance level, explicitly described by a user. A *Preference Assertion* represents a requirement engineer's preference, defined by propositional predicates, as described in Table 1. Thus, consider $T = \{t_1, t_2, t_3, ...t_Z\}$ the set of all preferences, where Z is the number of preferences. Each t_i is a tuple which contains the corresponding preference assertion and the importance level $L_i \in [1, 10]$. This modeling is provided to favor the process of preferences manipulation.

2.3 The Interactive Formulation for Software Release Planning

Considering the definitions in Sects. 2.1 and 2.2, the fitness function is defined as:

$$Fitness(S) = \begin{cases} score(S), & if\ Z = 0 \\ \frac{score(S)}{penalty(S)} & otherwise \end{cases}$$

where $score(S)$ is defined as:

$$score(S) = \sum_{i=1}^{N} y_i \times (value_i \times (P - x_i + 1) - risk_i \times x_i)$$

where $y_i \in \{0, 1\}$ is 1 if requirement r_i was allocated in some release, that is, $x_i \neq 0$, and 0 otherwise. The $value_i$ contains the weighted sum of importance specified by each client c_j for a requirement r_i, calculated by:

$$value_i = \sum_{j=1}^{M} w_j \times importance(c_j, r_i)$$

Therefore, the $score(S)$ function is higher when the requirements with highest $value$ and $risk$ are allocate in earlier releases.

When there are preferences, which are obtained by user interaction, the $Fitness(S)$ is penalized according to the importance level of each preference which was not satisfied, as follow:

$$penalty(S) = 1 + \mu \times \left(\frac{\sum_{i=1}^{Z} L_i \times violation(S, T_i)}{\sum_{i=1}^{Z} L_i} \right)$$

where the parameter $\mu \in \mathbb{R}_0^+$ defines the weight of the user preferences in the penalty, L_i is the importance level of preference T_i and $violation(S, T_i)$ returns 0 if solution S satisfies the preference T_i and 1 otherwise. Therefore, the higher the number of not satisfied preferences the higher penalty value.

Thus, the proposed interactive formulation for release planning is:

$$\begin{aligned} &\text{maximize} && Fitness(S), \\ &\text{subject to} && \sum_{i=1}^{n} cost_i \times f_{i,q} \leqslant s_q, \forall q \in \{1, 2, ..., P\} \end{aligned}$$

Table 1. Set of preference assertions for Release Planning

Representation	**Basic Interpretation**
Arguments	**Formal Interpretation**
$coupling\_joint(r_i, r_j)$	Two distinct requirements should be placed in the same release.
Requirements r_i and r_j.	$coupling\_joint(r_i, r_j)$ is $satisfied$ iff $x_i = x_j$.
$coupling\_disjoint(r_i, r_j)$	Two distinct requirements should be placed in different releases.
Requirements r_i and r_j.	$coupling\_disjoint(r_i, r_j)$ is $satisfied$ iff $x_i \neq x_j$.
$positioning\_precedes(r_i, r_j, [distance])$	One requirement should precede another by some distance.
Requirements r_i, r_j and a distance between requirements ($[distance]$), always higher than zero.	$positioning\_precedes(r_i, r_j, [distance])$ is $satisfied$ if at least one of the following conditions is met: i x_i, $x_j \neq 0$, $x_j - x_i \geq distance$, ii $x_i \neq 0$, $x_j = 0$.
$positioning\_follows(r_i, r_j, [distance])$	One requirement should follow another by some distance.
Requirements r_i, r_j and a distance between requirements ($[distance]$), always higher than zero.	$positioning\_follows(r_i, r_j, [distance])$ is $satisfied$ if at least one of the following conditions is met: i x_i, $x_j \neq 0$, $x_i - x_j \geq distance$, ii $x_i = 0$, $x_j, \neq 0$.
$positioning\_after(r_i, k_q)$	One requirement should be placed after a certain release.
Requirement r_i and a release $k_q \neq 0$.	$positioning\_after(r_i, k_q)$ is $satisfied$ if at least one of the following conditions is met: i $x_i \neq 0$, $x_i - k_q \geq 1$, ii $x_i = 0$.
$positioning\_before(r_i, k_q)$	One requirement should be placed before a certain release.
Requirement r_i and a release $k_q \neq 0$.	$positioning\_before(r_i, k_q, [distance])$, is $satisfied$ iff $x_i \neq 0$, $k_q - x_i \geq 1$
$positioning\_in(r_i, k_q)$	One requirement should be placed in a certain release.
Requirement r_i and a release $k_q \neq 0$.	$positioning\_before(r_i, k_q)$ is $satisfied$ iff $x_i = k_q$.

where $f_{i,q}$ indicates whether the requirement r_i was allocated in the release k_q.

3 Preliminary Empirical Study

A preliminary empirical study was conducted to evaluate the proposed approach over two distinct instances composed by real data with 50 and 25 independent requirements obtained from [8], named as dataset-1 and dataset-2, respectively.

The implementation risk of each requirement was randomly assigned. The number of releases for dataset-1 and dataset-2 was fixed to 5 and 8, respectively. The budget for each release was defined as the sum of all requirements costs divided by the number of releases. The instances and results are available on-line[1].

Regarding to the search algorithm, the IGA was applied with 100 individuals per population, 1000 generations, 90 % crossover rate, 1 % mutation rate and 20 % elitism rate. These parameters were empirically obtained. The IGA was executed 30 times for each instance and μ variation.

To simulate a user, for each instance, a set of preference assertions, without conflicting, was randomly generated and included in the *Preferences Base*. The number of preferences was 50 and 25 for the dataset-1 and dataset-2, respectively.

The experiments aimed at answering the follow research question:

RQ: How effective is the approach in finding solutions which satisfy a high number of important preferences?

3.1 Results and Analysis

Table 2 shows average and standard deviation for the percentage of number of *Satisfied Preferences* (*SP*), *Satisfaction Level (SL)* and *score* values of the solution for each instance when μ varies. *SL* is a percentage of how much was reached of the total importance of all preferences.

Table 2. Results of *SP*, *SL* and *score* with μ variation for each instance. The symbol $\triangle$ means this result is not significantly higher than the previous one, considering the μ variation, $\triangledown$ (not significantly lower), ▲ (significantly higher) and ▼ (significantly lower), considerering a 0.05 significance level

μ	dataset-1			dataset-2		
	SP	*SL*	*Score*	*SP*	*SL*	*Score*
0	0.40±0.03	0.40±0.02	25074.8±58.33	0.37±0.05	0.36±0.05	38561.3±154.8
0.1	0.54±0.01 ▲	0.57±0.02 ▲	24889.8±80.55 ▼	0.58±0.03 ▲	0.58±0.04 ▲	38359.9±168.4 ▼
0.2	0.62±0.02 ▲	0.66±0.02 ▲	24591.2±104.23 ▼	0.64±0.04 ▲	0.66±0.04 ▲	37871.7±425.9 ▼
0.3	0.65±0.02 ▲	0.71±0.02 ▲	24312.2±152.30 ▼	0.71±0.05 ▲	0.73±0.05 ▲	37218.1±583.9 ▼
0.4	0.74±0.02 ▲	0.77±0.03 ▲	23862.6±292.98 ▼	0.73±0.04 ▲	0.76±0.05 ▲	36954.5±624.9 ▼
0.5	0.75±0.03 ▲	0.80±0.02 ▲	23568.0±270.29 ▼	0.77±0.05 △	0.81±0.05 △	36332.8±646.7 ▽
0.6	0.77±0.02 ▲	0.83±0.02 ▲	23173.3±288.83 ▼	0.80±0.03 ▲	0.85±0.05 ▲	35774.0±873.3 ▼
0.7	0.80±0.03 △	0.86±0.02 △	22867.4±315.07 ▽	0.83±0.04 △	0.88±0.05 △	35211.4±999.6 ▽
0.8	0.81±0.02 △	0.87±0.01 ▲	22804.4±287.04 ▼	0.86±0.05 ▲	0.91±0.04 ▲	34630.7±902.4 ▼
0.9	0.82±0.02 ▲	0.87±0.01 △	22731.9±315.73 ▽	0.86±0.04 △	0.93±0.03 △	34459.7±802.9 ▽
1	0.83±0.02 △	0.88±0.01 △	22494.3±477.97 ▽	0.88±0.04 △	0.94±0.04 △	34052.5±674.0 ▽

With $\mu = 0$, that is, without considering the user preferences during the search process, the solutions satisfied in average 40 % and 37 % of all preferences, reaching 40 % and 36 % of *SL* respectively for dataset-1 and dataset-2. Using $\mu = 0.2$,

[1] http://goes.uece.br/altinodantas/pb4isrp/en.

SP reached 62 % and *SL* raised to 66 % for dataset-1, 64 % and 66 % for dataset-2. Comparing the results from $\mu = 1$ to $\mu = 0$, dataset-1, *SP* and *SL* increased 43 % and 48 % respectively, with a scoring loss of only 10.3 %. For dataset-2, the increments were 51 % and 58 % and score loss of 11.7 %.

So, given the number of user preferences equals to the number of requirements, it is possible to satisfy more than 80 % of preferences and get about 90 % of *Satisfaction Level* losing a maximum of 11.7 % of *score*. Therefore, these results answer the *RQ*, showing that the approach can satisfy the most the preferences with high importance level. Besides, *Wilcoxon Test* showed that, for lower values of μ, there was significant increase in *SP* and *SL*, specially, but, with significant loss of *score*. For values of μ near 1, there was no significant variations in *SP*, *SL* and *score*. These results can indicate the more appropriate μ configuration.

4 Conclusions

In any iterative software development process, the decision about which requirements will be allocated in each software release is as complex task.

The main objective of this work was to propose an interactive approach using a preferences base for release planning. An IGA was employed, guided by a *Preferences Base*, which provided a final solution able to satisfy almost of all user preferences, prioritizing the most important ones, with little loss of *score*.

As future works, it is expected to implement a mechanism to identify logical conflicts between user preferences; assess the proposal with other interactive metaheuristics and consider interdependences between requirements.

References

1. Ruhe, G., Ngo-The, A.: A systematic approach for solving the wicked problem of software release planning. Soft. Comput. **12**(1), 95–108 (2008)
2. Ruhe, G., Saliu, M.O.: The art and science of software release planning. IEEE Softw. **22**(6), 47–53 (2005)
3. Takagi, H.: Interactive evolutionary computation: fusion of the capabilities of ec optimization and human evaluation. Proc. IEEE **89**(9), 1275–1296 (2001)
4. Harman, M., Mansouri, S.A., Zhang, Y.: Search based software engineering: a comprehensive analysis and review of trends techniques and applications. Department of CS, King College London, Technical report. TR-09-03 (2009)
5. Greer, D., Ruhe, G.: Software release planning: an evolutionary and iterative approach. Inf. Softw. Technol. **46**(4), 243–253 (2004)
6. Colares, F., Souza, J., Carmo, R., Pádua, C., Mateus, G.R.: A new approach to the software release planning. In: XXIII Brazilian Symposium on Software Engineering, SBES 2009, pp. 207–215. IEEE (2009)
7. Araújo, A.A., Paixão, M.: Machine learning for user modeling in an interactive genetic algorithm for the next release problem. In: Le Goues, C., Yoo, S. (eds.) SSBSE 2014. LNCS, vol. 8636, pp. 228–233. Springer, Heidelberg (2014)
8. Karim, M.R., Ruhe, G.: Bi-objective genetic search for release planning in support of themes. In: Le Goues, C., Yoo, S. (eds.) SSBSE 2014. LNCS, vol. 8636, pp. 123–137. Springer, Heidelberg (2014)