Autom Softw Eng (2017) 24:623–671
DOI 10.1007/s10515-016-0200-3

An Architecture based on interactive optimization and machine learning applied to the next release problem

Allysson Allex Araújo[1] · Matheus Paixao[2] · Italo Yeltsin[1] · Altino Dantas[1] · Jerffeson Souza[1]

Received: 1 April 2015 / Accepted: 20 May 2016 / Published online: 6 June 2016

Abstract The next release problem (NRP) consists of selecting which requirements will be implemented in the next release of a software system. For many search based software engineering approaches to the NRP, there is still a lack of capability to efficiently incorporate human experience and preferences in the search process. Therefore, this paper proposes an architecture to deal with this issue, where the decision maker (DM) and his/her tacit assessments are taken into account during the solutions evaluations alongside the interactive genetic algorithm. Furthermore, a learning model is employed to avoid an overwhelming number of interactions. An empirical study involving software engineer practitioners, different instances, and different machine learning techniques was performed to assess the feasibility of the architecture to incorporate human knowledge in the overall optimization process. Obtained results indicate the architecture can assist the DM in selecting a set of requirements that properly incor-

✉ Allysson Allex Araújo
allysson.araujo@uece.br
http://goes.uece.br

Matheus Paixao
matheus.paixao.14@ucl.ac.uk
http://crest.cs.ucl.ac.uk

Italo Yeltsin
italo.yeltsin@uece.br

Altino Dantas
altino.dantas@uece.br

Jerffeson Souza
jerffeson.souza@uece.br

1 Optimization in Software Engineering Group, State University of Ceará, 1700, Dr. Silas Munguba Avenue, 60.714-903 Fortaleza, Brazil

2 CREST Centre, University College London, Malet Place, London WC1E 6BT, UK

porate his/her expertise, while optimizing other explicit measurable aspects equally important to the next release planning. On a scale of 0 (very ineffective) to 5 (very effective), all participants found the experience of interactively selecting the requirements using the approach as a 4 (effective).

Keywords Next release problem · Interactive optimization · Machine learning · Search based software engineering

1 Introduction

During an iterative and incremental software development process, there are some complex decisions to be made. Selecting which requirements will be implemented in the next release of the system is one of them, given the high number of combinations, technical constraints, multiple objectives and different stakeholders. "Release" is the term used to describe a stable and executable version of the system, which is delivered according to stakeholders requests. Given this context, the problem of maximizing the stakeholders satisfaction, while respecting a predefined budget for the release development, is called the next release problem (NRP) (Bagnall et al. 2001). The stakeholder is usually satisfied according to the requirements he/she wants the most, which are implemented in the next release.

The main goal of the search based software engineering (SBSE) field is to reformulate difficult problems found in software engineering into search problems. Then, the problems are solved by the use of computational search, especially metaheuristics. These search techniques are guided by a fitness function, which is able to distinguish good solutions from not so good ones (Harman 2007a). Thus, SBSE needs only two key ingredients: (i) the choice of the representation of the problem and (ii) the definition of the fitness function (Harman et al. 2012).

State of the art single objective SBSE approaches to the NRP usually select the requirements for the next release in a fully automatic fashion, without an effective and dynamic participation of the decision maker (DM). Most of the approaches assume that all the required information is collected before the optimization process. However, asking the DM to express a priori his/her knowledge might be an inconvenient task to perform, especially because there is too much information to consider, where most of it is actually implicit (Ferrucci et al. 2014).

Moreover, when the DM is not employed in the resolution process, he/she may feel excluded from the analysis, and present some resistance or put little confidence in the final result (Miettinen 1999,Shackelford 2007). In addition, there are several intrinsic aspects of requirements engineering which comprehension is inherently subjective, demanding a more effective user collaboration (Harman 2007b). Therefore, a user-friendly inclusion of the human expertise during the search process can result in a better strategy to handle the NRP, providing a decision support tool that is able to deal with all these matters and find a solution that successfully incorporates the DM's qualitative assessments.

The interactive evolutionary computation (IEC), which is a branch of interactive optimization, is supported by two key components: (i) human evaluation and

(ii) computational search through bio-inspired evolutionary strategies (Takagi 1998). Population-based algorithms seem to be ideal for interactive optimization since a user can directly evaluate fitness values of potential solutions (Takagi 2001). This optimization strategy is often necessary when the algorithm that defines the fitness function is either too complex to specify or, in many cases, impossible to quantify and code (Shackelford 2007).

Despite being widely spread in SBSE, currently used genetic algorithms (GAs) still lack the capability of efficiently considering human expertise in the search process. Thus, the interactive genetic algorithm (IGA) emerges as an interesting alternative in order to include the DM in the GA evolution process. The IGA shares several key concepts with a traditional GA, such as population evolution by applying selection, crossover and mutation operators. Differently, the solution evaluation is performed considering the DM's knowledge, guiding the search process to more valuable areas in subjective terms (Cho 2002).

Although an intense human involvement is interesting and attractive to the search process, it may cause one of the most critical drawbacks of interactive optimization approaches: the human fatigue (Takagi 2001). This exhaustion occurs due to repeated requests for human judgements, which ends up distracting the focus of the user over the algorithm execution, then jeopardizing the search trajectory (Simons et al. 2014). Dealing with this intrinsic complication can be considered a mandatory feature for any interactive optimization approach (Shackelford 2007).

Based on Kamalian et al. (2006), this paper handles the human fatigue using a machine learning model. The DM's expertise is gathered when he/she provides subjective evaluations to each solution during the algorithm evolution. Then, after the creation of a training dataset composed by different releases and respective human evaluations, a learning model can be used to learn the human behavior and, eventually, replace the DM in the remainder of the evolutionary process.

An initial proposal of an architecture that allows the DM to take part in a search-based approach to the NRP has been introduced in Araújo et al. (2014). In this first work, a conceptual version of the architecture was presented and empirically evaluated. Preliminary results were able to show that an IGA can successfully incorporate the DM preferences in the final solution. Therefore, this paper has the main objective of thoroughly present and evaluate the architecture to solve an interactive version of the next release problem that allows an inclusion of human knowledge during the search process using an interactive genetic algorithm.

The present paper significantly extends the previous work in three different aspects: (a) the learning model, previously only proposed as an idea, is now developed and considered in the empirical evaluation of the approach; (b) the empirical evaluation considering artificial cases was extended with two new research questions. Furthermore, the approach has also been evaluated with software engineering practitioners; and (c) the description of the architecture has been improved and extended through the definition of a new component, called the Interactive Module. The primary contributions of this paper can be summarized as:

– The presentation of an interactive single objective formulation of the NRP, called iNRP, that suits the explained architecture;

– A suite of proposed metrics that may contribute to the SBSE research community for the evaluation of empirical studies involving interactive optimization;
– Analyses of the behavior of the architecture when considering both simulated and real human subjective evaluations.

As previously mentioned, two experiments were performed in the empirical study conducted in this work, respectively named as: (a) artificial experiment and (b) participant-based experiment. In the first one, a simulator was employed to verify the influence of different evaluation profiles in several and exhaustive scenarios, including both artificial and real-world instances. In the second experiment, a group of software engineer practitioners was invited to solve a real-world instance using the proposed approach. For each experiment, two learning techniques were used based on different strategies of machine learning.

The remainder of this paper is organized as follows: Sect. 2 explains the proposed architecture; Sect. 3 presents the interactive formulation for the NRP, while Sect. 4 exhibits and examines the empirical study designed to evaluate the proposal; Sect. 5 discusses some work related to this paper; finally, Sect. 6 concludes the paper and points out some future research directions.

2 Architecture overview

Domain knowledge can be used to guide the optimization algorithm to promising areas of the search landscape. Such an approach enables the search to simultaneously and effortlessly adapt the solution to the business needs of the DM, while reducing the size of the search space. It is recommended to: "wherever possible, domain knowledge should be incorporated into the SBSE approach" (Harman et al. 2012). However, the design of search approaches that are able to naturally incorporate the subjective preferences into the overall optimization process is still a challenge for the SBSE community.

Following the definitions stated in Miettinen et al. (2016), the role of the DM in the solution process can be divided into four classes: (i) no-preference methods in which there is no need of articulation of preference information; (ii) a priori methods that require articulation of preference before the search process; (iii) a posteriori methods in which the preference information is used after the search process and, finally, (iv) interactive methods in which the DM introduces preference information progressively during the search process. This proposal focuses on interactive methods.

One important assumption of this work is that, regardless of how many stakeholders are involved in the project, there will be a specific DM to make the last call. This decision is motivated by the need of an unbiased judgement that is able to manage the possible conflicts of interests of the other stakeholders and also weight the different characteristics of the decision-making process. Such a role should be performed by a professional fully immersed in the high-level project particularities and with a broad view of his/her short/long time decisions. Hence, the presented approach focuses in interacting with this only one crucial stakeholder, referred in this work as decision maker (DM).

After deciding how the human will provide its preferences and who will be the DM in the software project, the next step is to understand the nature of human preference. According to the work in Aljawawdeh et al. (2015), the preferences can be categorized as explicit and implicit. Explicit preferences are readily articulated by the DM and their relevance to the search is typically well understood by him/her, while implicit preferences may be tacit and difficult for humans to previously articulate. As an example, implicit memory is a type of memory that previous experiences might aid in the performance of a task without conscious awareness of these experiences (Schachter 1987).

Therefore, this paper proposes a conceptual architecture that is able to interactively incorporate in a step-wise manner both explicit and implicit preferences during the search process through an interactive genetic algorithm. To overcome the need of human intervention in all individuals evaluations, a machine learning model is used to learn the user's profile and, consequently, be able to replace him/her when necessary alongside the IGA. Such architecture (presented in Fig. 1), is composed of three main components, which are generically described below:

1. **Interactive genetic algorithm** responsible for the optimization process, where the candidate solutions are firstly evaluated by the Interactive Module and, when required, by the learning model. Given an individual, its fitness value is calculated considering both explicit and implicit preferences. Explicit preference is represented by well defined aspects of the problem, such as the quantitative measures of $score$ value and the implementation effort of each requirement, as addressed by the classical NRP (Bagnall et al. 2001). Implicit preference is gathered by a qualitative assessment of the DM regarding a certain solution, which is formally defined as subjective evaluation (*SE*). The transition between using the Interactive Module and the Learning Model is dynamically performed depending on how many interactions the DM is willing to perform.
2. **Interactive module** responsible for the interactive interface with the DM. Thus, each *interaction* consists of an individual of the IGA that receives a *SE* value from the DM. The *SE* value of that respective individual is passed to the IGA, while both the individual and its *SE* value are passed to the learning model. The only constraint between the number of interactions and individuals is to have a number of individuals that is, at least, equal or bigger than the number of interactions. Thus, it is possible to have more individuals than interactions, but not vice-versa.
3. **Learning model** responsible for learning the user profile through a machine learning technique, in which all details regarding the learning problem formulation and representation have to be well-formalized. The training process occurs in this component, according to the samples (individuals and *SE* values) provided by the Interactive Module. After the training process be performed, the learning model will be able to provide a *SE* value to each solution that needs to be subjectively evaluated. Naturally, the *SE* value provided by the learning model will only be suitable for the learnt DM's profile. In addition, the effectiveness of the learning model will be proportional to the number of samples provided by the Interactive Module.

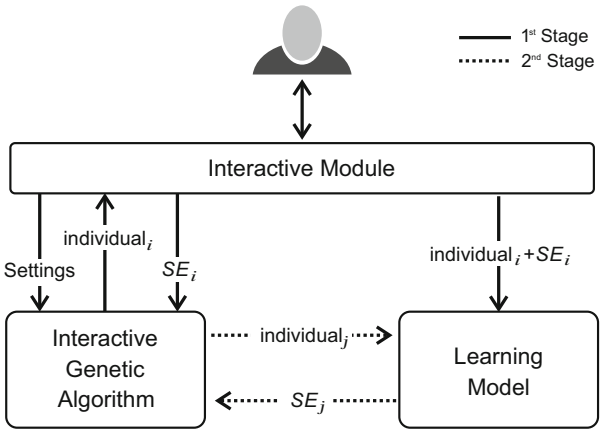

Fig. 1 Conceptual architecture overview

As presented in Fig. 1, the relationship between these components are divided in two distinct stages, defined below:

(a) **First stage (solid lines)** before starting the search process, it is necessary for the DM to specify two architectural settings: (i) the weight of the implicit preferences in comparison to the explicit ones for the fitness calculation. This parameter is related to how influential the DM's subjective knowledge will be in the search process; (ii) a minimum number of interactions i which the DM is willing to take part in. This parameter defines the first i individuals to be evaluated by the DM, where each $individual_i$ and its respective SE_i value are sent to the learning model.

(b) **Second stage (dotted lines)** at this moment, the learning process is performed using the set of samples collected in the previous stage as a training dataset. After the conclusion of this process, the learning model will be responsible for simulating the DM behavior and evaluates the remainder of j individuals from the evolutionary process, in other words, providing a SE_j value for each $individual_j$ until the stopping criteria is reached. At the end, the best solution found by the IGA is presented to the DM.

Figure 2 presents an activity diagram that shows the overall optimization process, focusing on how the different components of the architecture exchange information. As explained before, the IGA primarily follows the concepts of a traditional GA, with the primary difference being the fitness evaluation. As depicted in the Fig. 2, after the architectural settings definition (weight of the subjective evaluation and number of interactions), the population is initialized and the stopping criteria is verified. While the criteria is not reached, the search process continues.

As previously mentioned, the fitness value of an individual considers both explicit preferences and the *SE* value that encapsulates the implicit preference of the DM. The decision to stop requiring *SE* values from the Interactive Module and changing to receive *SE* values from the learning model is defined according to the minimum number of interactions i established by the DM. Therefore:

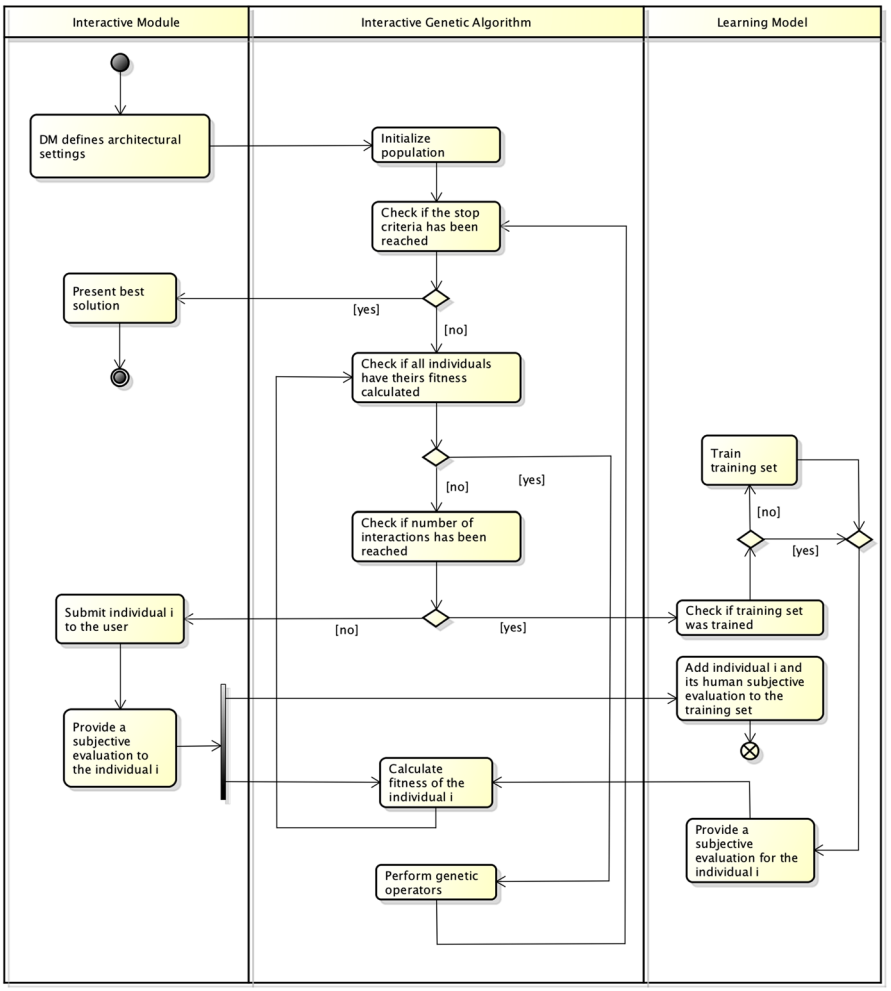

Fig. 2 Activity diagram of the conceptual architecture

– If the number of interactions *has not* been reached, the individual is presented to the DM, which will be responsible for giving a SE. This evaluation is then considered in the fitness calculation of $individual_i$, and the SE_i value alongside its respective $individual_i$ are included in the training dataset for the learning model.
– If the number of interactions *has* been reached, it is checked whether the learning model has been trained. If not, the training process is performed considering the samples collected in the previous stage as training dataset. Then, the learning model provides a SE_j for the $individual_j$ to be evaluated. In the case where the learning model has already been trained, it directly provides a SE_j for each presented $individual_j$.

When the fitness of all individuals in the population are calculated, the genetic operators are applied and, consequently, the algorithm carries on considering the subjective

evaluations provided by the learning model until the stopping criteria is reached. Upon finishing, the best solution achieved is shown to the DM.

This conceptual architecture can be considered sufficiently generic to be adopted in other software engineering scenarios tackled by SBSE, such as feature selection in software product lines, requirements prioritization or software design problems. However, this paper is focused on evaluating the approach to the NRP, in which all formalization required to be suited to the architecture is properly presented in the next section.

3 An interactive next release problem formulation

In SBSE, the fitness function guides the search by capturing the properties that make a given solution preferable to another one (Zhang et al. 2008). Usually, the fitness function is composed of metrics that represent quality and constraint attributes from the software asset being measured and optimized (Harman and Clark 2004). Unfortunately, sometimes these attributes may not be precisely defined in the early stages of the software life cycle, or are inherently associated with comprehension activities, which require a human input to the assessment (Harman et al. 2012). This kind of problem can be properly handled by algorithms that employ a "human-in-the-loop" fitness computation, in other words, interactive optimization.

As previously mentioned, in order to apply a search based approach to a Software Engineering problem, one needs the solution representation and fitness function (Harman 2007a). However, since this paper proposes an architecture based on SBSE, interactive optimization and machine learning foundations, three major questions had to be properly defined: how the objectives to be optimized are mathematically modelled? How does the DM interact with the optimization process? And which details are required to allow the learning process to be performed? Therefore, the problem formulation in this work is composed of a triad of formulations, respectively named as mathematical modeling, interactive modeling and learning modeling.

3.1 Mathematical modeling

Consider $R = \{r_1, r_2, r_3, \ldots, r_N\}$ the set of all available requirements to be selected for the next release, where N is the maximum number of requirements. Each requirement r_i has an importance value v_i and an implementation effort e_i. Consider $K = \{k_1, k_2, k_3, \ldots, k_M\}$ as the set of stakeholders to be attended by the system, where each stakeholder k_j has a specific weight w_j that measures his/her relevance to the software project. The interactive next release problem (iNRP) model proposed in this work can be formalized as follows:

$$maximize\ \alpha \times score(X) + \beta \times SE(X), \tag{1}$$

$$\text{subject to: } effort(X) \leq budget, \tag{2}$$

$$\text{where, } score(X) = \sum_{i=1}^{N} v_i \times x_i, \tag{3}$$

$$v_i = \sum_{j=1}^{M} w_j \times s_{ji}, \tag{4}$$

$$effort(X) = \sum_{i=1}^{N} e_i \times x_i, \tag{5}$$

where $budget$ refers to the release available budget. A binary solution representation is used, where the release is represented by the decision variables $X = \{x_1, x_2, \ldots, x_N\}$, so that $x_i = 1$ implies that requirement r_i is included in the next release and $x_i = 0$ otherwise. The $score(X)$ function (Eq. 3) is calculated by multiplying the sum of the total importance (v_i) of requirement r_i and the decision variable x_i. The total importance (v_i) of a requirement r_i is given by the product of the weight (w_j) of each stakeholder (k_j) and the specific importance (s_{ji}) for the requirement r_i provided by each stakeholder (see Eq. 4). Generally, the $score(X)$ function encourages the search to achieve solutions that maximize the overall stakeholders satisfaction. Similarly to the $score(X)$ function, the $effort(X)$ function (Eq. 5) represents the total implementation $effort$ of the release and it is calculated by the product of the sum of each requirement $effort(X)$ (e_i) and the decision variable x_i. As a constraint, the $effort(X)$ cannot exceed the $budget$. Therefore, this formulation provide the quality of a candidate solution, also known as, the fitness (F).

In the IGA application to the iNRP, each individual is a release. However, as explained in the Sect. 2, two kinds of information are needed to calculate the fitness of an individual: the explicit and the implicit preferences. The first one is represented by the $score(X)$ objective obtained using Eq. 3, which represents the overall stakeholders satisfaction, while the second is encapsulated by the DM's tacit assessment obtained from $SE(X)$. The implicit preference provided by the DM was modeled using a numerical range value established as a qualitative "grade". So, when the release fully satisfies the DM, the grade is maximum. Similarly, the grade is minimal when the release is completely different from what the DM expects. Thus, the search process will be guided to areas that maximize both the $score(X)$ values and the DM's satisfaction.

To avoid any misconceptions, this paper considers "explicit preference" as the result of the $score(X)$ function and the "implicit preference" as the subjective evaluation provided by the DM ($SE(X)$). Thus, in some cases, the DM opinion towards the selected requirements is not influenced by the explicit one, because the importance assigned by a specific stakeholder to a certain requirement may not agree with the DM strategic planning. So, it is reasonable to expect some trade-off regarding the loss in the overall stakeholders satisfaction to achieve a better solution in subjective aspects. Later on, it will be explained how this trade-off can be properly balanced according to the specific needs of a certain software project.

Another important aspect to take into account under the context of next release planning are the interdependencies between requirements. This paper considers the functional interdependencies REQUIRE(r_i, r_j) and AND(r_i, r_j) (Carlshamre et al. 2001). These two interdependencies were chosen because they are the most common in software projects. A REQUIRE(r_i, r_j) dependency indicates that requirement r_i cannot be selected to the next release if r_j is not selected as well. However, requirement

r_j can be included in the next release without r_i. Differently, AND(r_i, r_j) indicates that both requirements should be selected for the release, otherwise none of them can. Both interdependencies are represented by a single matrix $N \times N$ called "functional". Thus, $functional_{ij} = 1$ indicates the existence of a dependency REQUIRE(r_i, r_j), while a dependency AND(r_i, r_j) is represented by $functional_{ij} = functional_{ji} = 1$.

The mathematical model used in this paper can be considered as a generalization of the one proposed in Baker et al. (2006). When the weights α and β in Eq. 1 are configured to $\alpha = 1$ and $\beta = 0$, the classical NRP is reached. In other words, the requirements selection will consider only the $score(X)$ function (Eq. 3). When the weights are configured to $\alpha = 0$ and $\beta = 1$, only the subjective evaluations ($SE(X)$) will be considered in the search process. The $\alpha = 1$ and $\beta = 1$ configuration consider the DM's assessment and, at the same time, select the requirements with highest $score(X)$. This freedom of choice to be determined before the algorithm execution allows the proposed approach to be adapted so specific needs of different software projects. For example, if it is established to prioritize the DM's subjective evaluations over the stakeholders satisfaction, one can simply choose a $\alpha = 1$ and $\beta = 2$ configuration. On the other hand, if it is decided to prioritize the importance values assigned by the stakeholders more than the DM's strategic opinions, a $\alpha = 2$ and $\beta = 1$ configuration can be employed. Moreover, the fitness calculation addressed in this formulation is summarized by the following algorithm:

Algorithm 1: Fitness calculation of a certain individual

Input: individual, SE, α, β
Output: Fitness of individual
begin
  Calculate $score$ of individual
  Normalize $score$ to the same range of SE
  Fitness of individual $\leftarrow (\alpha \times score) + (\beta \times SE)$
end

In this work, it was opted to normalize the $score(X)$ value to the same interval of $SE(X)$, in order to avoid a possible unbalance among these values during the optimization process. Given this context, the only method to prioritize one function over the other is by assigning values to the weights α and β in the fitness function. The $score(X)$ normalization is described as follows:

$$normalizedScore(X) = \left(\frac{score(X)}{score_{max}}\right) \times SE_{max}, \tag{6}$$

where, X represents a solution, $score_{max}$ is the highest value in terms of $score$ which a solution can have and SE_{max} is the highest grade which the subjective evaluation can assume.

3.2 Interactive modeling

There have been several studies that explore Interactive Optimization in SBSE, as will be discussed in Sect. 5.2. In recent years there has also been an increase in the number

of empirical studies involving interactive optimization application to the Software Engineering. As stated in Glass (2002), "the most important factor in software work is not the tools and the techniques used by the programmers, but rather the quality of the programmers themselves".

However, there is a challenge in designing suitable environments to exploit these human qualities in the search process, given that the user knowledge can be expressed in a variety of forms. This paper considers the understanding of the whole process regarding the human interaction and his/her influence in the optimization process to be a very important issue. As discussed in Aljawawdeh et al. (2015), the nature of the implicit preference and the moment in which this aspect will be exploited during the optimization process are the main aspects to be modelled in an interactive optimization approach.

Therefore, three main questions are proposed and, consequently discussed, to define the Interactive Modeling for the iNRP.

1. **At which moment are the preferences from the DM captured?**
   As previously discussed, the moment of preference incorporation in the search process ranges from no-preference, a priori, a posteriori and interactively (Miettinen et al. 2016). To the present proposal, the DM will interactively provide subjective evaluations for a previously defined number of individuals generated by the IGA. This number of interactions is a architectural setting defined before to start the search process, which depends on DM availability. Given a scenario delimited by 50 interactions and a population with 50 individuals, for example, all the 50 individuals from the first generation of the IGA will be evaluated by the DM. On the other hand, from the first individual of the second generation until the end of the IGA, the individuals will be evaluated by the learning model. Similarly, in a hypothetical case of 50 human interactions and 25 individuals, the first two generations of the IGA will have all individuals evaluated by the DM, and the Learning Model will start evaluating individuals from the third generation and on. At the end of the process, the best final solution is presented to the DM.
2. **What type and which preferences are provided to the search process?**
   Following the assumptions detailed in Aljawawdeh et al. (2015), the nature of human preferences are explicit or implicit. As stated before, this work explores both types, being "explicit" the result of the objective measure $score$ function and "implicit" the subjective evaluation provided by the DM (SE). The first one reflects the overall stakeholders satisfaction, while the second encapsulates several of the DM's subjective concerns regarding the presented solution. This value must be within a predefined range representing "how good" he/she thinks that selection is. The DM is not actually looking for a precisely defined numerical target but rather for a subset of the search space that gives the general impression he/she is looking for. The fuzzy aspect of human subjectivity is therefore more to be taken as a basis for robustness than as a source of trouble (Semet 2002).
3. **How the preferences are incorporated and influence the search process?**
   Human preferences can be incorporated and influence the search process in several ways. In Aljawawdeh et al. (2015) six potential design pattern abstractions considering the combination between the moment and type of preferences (explicit and

implicit) are presented. One of them inspired this work, and it is called "Implicit preference, interactive", in which the definition states: "users are offered opportunities to input preference to metaheuristic search either as qualitative evaluation solely or in combination with quantitative objective fitness functions". In the present study, the subjective evaluation will be exploited as an objective alongside the explicit metric $score$ in the fitness function, guiding the search to areas which maximizes both the DM's evaluations and $score$ values. In addition, as will be further discussed in the Related Work (Sect. 5), there are other ways to incorporate the DM's opinion in the search process beyond being an objective to be maximized.

The Interactive Modeling described above enables the DM to incorporate his/her knowledge during the search process. Generally, three interesting benefits arise from the usage of Interactive Evolutionary Computation:

(a) As the DM expresses his/her preferences during the evolutionary process, the DM will receive some feedback from the search through the presentation of the most promising solutions throughout the algorithm evolution. Since the solution fitness considers both implicit and explicit preferences (Aljawawdeh et al. 2015), it can be conjectured that, as the solutions are evaluated by the DM, there will be a natural trend for the new generations to be composed of solutions which better suits his/her subjective needs.
(b) Another interesting benefit that can be highlighted is the capability to incorporate the changes in the DM's criteria during the search process. As stated in Miettinen et al. (2016), "an important advantage of interactive methods is learning". Given the visualization of the solutions influenced by the DM opinion, he/she will progressively gain more consciousness of how attainable or feasible the preferences are. Consequently, he/she may get some insights about the problem, and even adapt the decision criteria. The proposed approach is able to deal with this aspect. This benefit reinforce one of the major goals of search based requirements optimization, that is to provide meaningful insights to the DM (Zhang et al. 2008). In the case of a change in the DM decision criteria using an a priori approach, the algorithm would have to be executed again to cope with the new preferences.
(c) At last, besides incorporating human knowledge, another important aspect of the interactive approach is human engagement. Intuitive interaction with domain specialists is a key factor in industrial applicability, since it makes the system more usable and more easily accepted in an industrial setting (Marculescu et al. 2015). As presented in the Introduction, when the DM is not employed in the resolution process, he/she may feel excluded in the analysis, which may cause resistance and lack of confidence in the final result (Miettinen 1999,Shackelford 2007).

3.3 Learning modeling

Machine Learning deals with the issue of how to build programs that improve their performance at some task through experience (Mitchell 1997). The field of Software Engineering turns out to be a fertile ground where many software development and maintenance tasks could be formulated as learning problems and approached in terms of learning algorithms (Zhang and Tsai 2003).

According to the work in Witten and Frank (2005), there are basically four different styles of learning. In *classification learning*, the learning scheme is presented with a set of classified examples from which it is expected to learn a way of classifying unseen examples. In *association learning*, any association among features is sought, not only the ones that predict a particular class value. In *clustering*, groups of examples that belong together are sought. In *numeric prediction*, the outcome to be predicted is not a discrete class but a numeric quantity.

This paper presents the idea of gathering the DM's implicit preferences during the algorithm evolution through subjectively evaluating a certain number of individuals, which in the NRP perspective are the releases. A machine learning model is used alongside the search to learn the human behavior and, eventually, replace it in the remainder of the process. In other words, classifying unseen examples following the learnt user profile. This strategy enables the system to absorb the user's implicit knowledge without requiring the user to formalize this information a priori (Shackelford 2007). However, as discussed in Zhang and Tsai (2003), user modeling poses a number of challenges for machine learning that have hindered its application in Software Engineering, including: the need for large data sets; the need for labeled data; concept drift; and computational complexity.

Similarly to the Interactive Modeling, two major questions are discussed aiming to define the learning modeling for the proposed iNRP:

1. **Problem formulation**
   An important first step is to formulate the problem in such a way that it conforms to the framework of a particular learning method chosen for the task. As stated in Mitchell (1997), "a computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E". Thus, to have a well-defined learning problem, one must identity these three features:
   – Task T: evaluate a release considering the human preferences;
   – Performance measure P: number of interactions;
   – Training experience E: evaluating releases provided throughout the optimization process.
2. **Problem representation**
   The next step is to define a representation for both training data and knowledge to be learned. The representation of the (a) *input*, (b) *attributes* and (c) *output* in the learning task is often problem-specific and formalism-dependent (Zhang and Tsai 2003.
   Following the concepts defined in Witten and Frank (2005), the *input* to a machine learning scheme is a set of samples. These samples are the things that need to be classified, associated or clustered. Each sample is an individual, independent example of the concept to be learned. The samples that provide the input to the machine learning model are characterized by its values on a fixed, predefined set of attributes. Also, it is important to highlight the presence of a special attribute called *class*, which describes the goal to be learned.
   Each training dataset is represented as a matrix of samples versus attributes, as depicted in Fig. 3. As can be seen in such figure, there are three samples, with

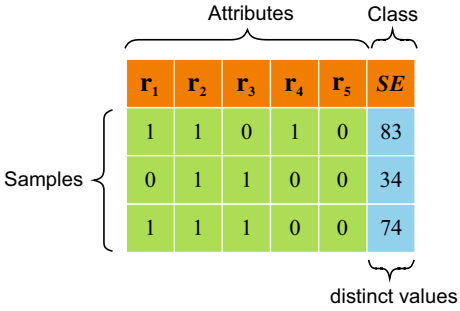

Fig. 3 Example of training dataset

each row representing a possible release and each column representing a specific requirement (attribute). The exception is the last column at the right, which represents the class obtained through the subjective evaluation provided by the DM. As explained in Section 3.1, $r_i = 0$ implies that requirement r_i is included in release and, $r_i = 0$ otherwise. For example, the first solution represented by $X = \{1, 1, 0, 1, 0\}$ is composed by requirements r_1, r_2 and r_4, and received a $SE(X) = 83$ from the DM. As explained in Sect. 2, this dataset will be iteratively constructed until the number of interactions defined by the DM in the architectural settings is reached. After concluding, the training process is performed considering the training dataset previously collected.

The *output* usually takes the form of predictions about new examples or classification of unknown examples. In the present context, the task is to evaluate releases according to the human preferences. Thus, considering that the Learning Model already learnt the user profile, the output will be a subjective evaluation to an unseen solution generated by the optimization process. As discussed above, both input and output follow a numerical range value established as a qualitative "grade". It is important to point out that the quality and the quantity of the data needed are dependent on both the selected machine learning technique and the number of provided training samples.

At last, an important aspect to be distinguished is the type of learning, which can be generally categorized as supervised and unsupervised. The first one is basically a synonym for classification, in which the learning will come from the labeled samples with a class in the training dataset, while the second is essentially a synonym for clustering, and is unsupervised since the input samples are not labeled (Han et al. 2011). Thus, the proposed learning model follows the supervised learning principles where each release is labeled with the respective class, defined as the SE.

4 Empirical study

The following sections present all the details regarding the empirical study. As previously mentioned, two experiments were performed, respectively named as (a) artificial experiment and (b) participant-based experiment. Firstly, the metrics developed to

evaluate the outcome of these experiments are explained. Next, general settings used in the experiments are presented, including the instances configurations, machine learning techniques and IGA parameters, specifications of each experiment and the research questions proposed to be answered. Then, with all these details being properly presented, the analyses and discussion of the achieved results are conducted. Finally, threats that may affect the validity of the experiments are also discussed.

4.1 Metrics

In order to promote meaningful analyses, three metrics were developed to clarify the results. It is believed such metrics are generic enough to be used in other research projects that explore the concepts of interactive optimization in SBSE.

4.1.1 Similarity degree

The *Similarity degree (SD)* indicates a percentage of how similar a candidate solution is when compared to the target solution. It is reasonable to consider that this metric directly represents a subjective satisfaction, given that the closer a candidate solution is from the target solution, the higher the subjective evaluation. Consider a set of 6 requirements with a target solution represented by $P = \{1, 0, 0, 0, 1, 1\}$ and a candidate solution represented by $X = \{1, 1, 0, 1, 1, 0\}$, for example. As one can see, x_1, x_3 and x_5 are equal in both P and X, thus, this candidate solution X would have a $SD(X, P) = \frac{3}{6} \times 100\,\% = 50\,\%$. This result is obtained by the following equation:

$$SD(X, P) = \left(\frac{\sum_{i|1 \leq i \leq N \wedge x_i = p_i}^{N} 1}{N} \right) \times 100\,\%, \tag{7}$$

where X is a candidate solution for which one wants to calculate its $SD(X, P)$ in relation to a target solution P. N is the number of requirements, x_i indicates whether the requirement r_i is present in the solution or not, and p_i indicates if the requirement r_i belongs to the target solution or not.

4.1.2 Similarity factor

The *Similarity factor (SF)* indicates the proportional gain in *SD* when comparing a solution with human influence and another one without human influence. For example, consider two solutions Y and X. The first solution with human intervention has a $SD(Y, P) = 85.33$, while The second one, without human influence, has a $SD(X, P) = 54.27$. Thus, the gain in *SF* achieved by Y over X is 57.2 %. This value is given by:

$$SF(Y, X, P) = \left(\frac{SD(Y, P)}{SD(X, P)} - 1 \right) \times 100\,\%, \tag{8}$$

where Y is the interactively generated solution, i.e., the search process is influenced by implicit preferences, X is the solution generated without considering subjective evaluations and P is the target solution.

4.1.3 Price of preference

The *Price of preference (PP)* shows how much is lost in explicit preference in order to incorporate implicit preferences from the DM through SE. Consider two solutions Y and X, for example. The first one generated with human influence has a $score(Y) = 99.89$, and the second one, without human influence, has a $score(X) = 115.87$. Therefore, the PP loss of Y over X is 16 %. This value is obtained by:

$$PP(X, Y) = \left(1 - \frac{score(Y)}{score(X)}\right) \times 100\,\%, \tag{9}$$

where Y is the solution considering subjective evaluations and X is the fully automatic solution, i.e., without any interaction.

4.2 Empirical study settings

This subsection initially presents the instances used in the experiments, including the number of requirements and interdependencies, number of stakeholders and budget definition. Then, the main concepts of the machine learning techniques employed in the tests are presented, as well as their respective parameters. The IGA parameters are also presented, alongside a brief discussion about the statistical techniques employed to analyze the obtained results.

4.2.1 Instances configuration

The instances set is composed with both artificial and real-world data. The artificial data was randomly generated and designed to represent different scenarios of next release planning. The number of requirements varies between 50, 100, 150 and 200. The specific importance of each requirement is ranged from a discrete value between 1 and 5. The number of stakeholders was randomly generated within a discrete range of 1 to 8. The weight of each stakeholder is given by a continuous random value between 0 and 1, so that the sum of the weights of stakeholders is equal to 1. The artificial instances name is in the format I_R, where R is the number of requirements. For example, if an instance has 50 requirements, it will be named I_50.

The real-world instances employed in this empirical study are the same in Karim and Ruhe (2014), respectively named as Word Processor and ReleasePlanner. The first one is based on a word processor software, being composed of 50 requirements and 4 customers. The second one is based on a decision support system and has 25 requirements and 9 customers. For all instances, including the artificial ones, the budget was considered to be 60 % of the maximum release cost.

Interdependencies between requirements present a considerable impact in the search space of the NRP Carlshamre et al. 2001). The interdependency density indicates the percentage of requirements that have dependencies in a particular instance. For the artificial instances, the interdependencies were randomly created at a density of 50 % following a procedure that is described next. Two requirements are randomly chosen, and the type of interdependency (REQUIRE(r_i, r_j) or AND(r_i, r_j)) they will have between themselves is also chosen at random. This process is repeated until the 50 % interdependency density is reached. For example, given a release with 100 requirements, 50 requirements will have at least one type of interdependency and each of these requirements will have a maximum of 50 dependents. The real-world instances have originally 82 and 40 % of interdependency density, respectively.

4.2.2 Machine learning techniques settings

With respect to the details regarding the learning process, the API provided by the Waikato Environment for Knowledge Analysis (WEKA) was emplyoed (Hall et al. 2009). WEKA is an open source platform that is characterized by a high degree of portability and modifiability. Aiming at having a more generic analysis, two techniques based on different learning strategies were chosen: least median square (LMS) and multilayer perceptron (MLP).

First of all, when the expected outcome and the attributes of a particular learning model are numeric, linear regression can be considered a natural technique to be used. Linear Regression performs standard least-squares multiple linear regression and can optionally perform attribute selection. Such feature is accomplished either by greedily using backward elimination or by building a full model from all attributes and dropping the terms one by one, in a decreasing order of their standardized coefficients, until a stopping criteria is reached (Witten and Frank 2005). The usual least-squares regression models are seriously affected by outliers in the data. As an attempt to minimize this issue, the least median square (LMS) is a robust linear regression method that minimizes the median (rather than the mean) of the squares of divergences from the regression line (Rousseeuw 1984). It repeatedly applies standard linear regression to subsamples of the data and outputs the solution that has the smallest median-squared error.

As defined in Witten and Frank (2005), the main idea is to express the class as a linear combination of the attributes, with predetermined weights:

$$x = w_0 + w_1 a_1 + w_2 a_2 + \cdots + w_k a_k, \tag{10}$$

where x is the class; $a_1, a_2, \ldots, a_k$ are the attribute values; and $w_0, w_1, \ldots, w_k$ are the weights. These weights are calculated from the training data. The first training sample will have a class, say $x^{(1)}$, and attribute values $a_1^{(1)}, a_2^{(1)}, \ldots, a_k^{(1)}$ where the superscript denotes that it is the first sample. Moreover, it is notationally convenient to assume an extra attribute a_0, with a value that is always 1.

The predicted value for the first example's class can be written as:

$$w_0 a_0^{(1)} + w_1 a_1^{(1)} + w_2 a_2^{(1)} + \cdots + w_k a_k^{(1)} = \sum_{j=0}^{k} w_j a_j^{(1)}, \tag{11}$$

The difference between the predicted and actual values for a certain sample is of particular interest for the LMS technique. The method of linear regression chooses the coefficients w_j to minimize the median of the sum of the squares of these differences over all the training samples. Suppose there are n training samples; denote the ith one with a superscript (i). Then, the sum of the squares of the differences is:

$$\sum_{i=1}^{n} \left(x^{(i)} - \sum_{j=0} w_j a_1^{(i)} \right)^2, \tag{12}$$

where the expression inside the parentheses is the difference between the ith samples's actual class and its predicted class. The median of the sum of squares is what has to be minimized through a properly selection of coefficients.

The multilayer perceptron (MLP) is a neural network that is trained using the backpropagation algorithm and, consequently, is capable of expressing a rich variety of nonlinear decision surfaces. Its main characteristics are: (a) the model of each neuron or network processing element has a nonlinear activation function; (b) it presents, at least, one intermediate layer that is not part of the input or output; and (c) it has a high degree of connectivity between its network processing elements, defined thorough the synaptic weights (Haykin 2001).

The backpropagation algorithm is able to learn the weights for a given multilayer network with a fixed set of units and interconnections. It is a version of the generic gradient descent, which attempts to minimize the squared error between the network output values and the target values for these outputs (Witten and Frank 2005). To use gradient descent to find the weights of a multilayer perceptron, the derivative of the squared error must be determined with respect to each weight in the network. According to Mitchell (1997), the gradient specifies the direction of the steepest increase of the error E, and the weight update rule is be given by:

$$w_i \leftarrow w_i + \Delta w_i, \tag{13}$$

where

$$\Delta w_i = \eta \sum_{d \in D} (t_d - O_d) x_{id}, \tag{14}$$

being n as a positive constant called the learning rate, which determines the step size in the gradient descent search. D is the dataset of training samples, t_d is the target output for training samples d, and o_d is the output of the linear unit for training samples d. Thus, x_{id} denotes the single input component x_i for a training samples d.

Table 1 Machine learning techniques settings

Learning technique	Parameter settings
Least median square	Sample size (-S): 4
	Used seed to generate samples (-G): 0
	Multilayer perceptron learning rate (-L): 0.3
	Momentum rate (-M): 0.2
	Number of epochs (-N): 500
Multilayer perceptron	percentage size of validation set (-V): 0
	The used value to seed the random number generator (-S): 0
	The consecutive number of errors allowed for validation, testing beforethe network terminates (-E): 20
	The hidden layers to be created for the network (-H): (attributes + classes) / 2

The work in Mitchell (1997) summarizes the gradient descent algorithm for training linear units as follows: pick an initial random weight vector, apply the linear unit to all training examples, then compute A_{wi} for each weight according to Eq. 14. Update each weight w_i by adding A_{wi}, then repeat this process.

Finally, Table 1 presents the machine learning techniques parametrization used in the empirical tests. More details regarding each parameter are available in Witten and Frank (2005).

4.2.3 Interactive genetic algorithm settings

All IGA's settings were empirically obtained by preliminary experiments. They are: number of individuals that is double of the number of requirements; 100 generations; 90 % crossover rate; $1/N$ mutation rate, where N is the number of requirements; 20 % of elitism rate.

To account for the inherent variation in stochastic optimization algorithms, for each weight configuration, number of interactions, instance and machine learning technique, the IGA was executed 30 times, collecting both quality metrics (*SD*, *SF* and *PP*) and respective averages from the obtained results. In the end, more than 55,000 executions of the IGA were performed.

Aiming at analyzing the results in a sound manner, guidelines suggested by Arcuri and Briand (2011) and the statistical computing tool R (R-Project 2016) were considered. Statistical difference is measured with the Mann–Whitney U test considering a 95 % confidence level, while the Vargha and Delaney's $\hat{A}_{12}$ test is used to measure the *effect sizes*. The $\hat{A}_{12}$ statistics measures the probability that a run with a particular algorithm 1 yields better values than a algorithm 2. This work assumes LMS_1 and MLP_2; therefore, $\hat{A}_{12} = 0.26$ in $score$, for example, indicates that, in 26 % of the time the LMS_1 reaches higher values than MLP_2. If there is no difference between two algorithms performances, then $\hat{A}_{12} = 0.5$. All instances and results of the empirical

study, including the ones that had to be omitted due to space constraints, are available online.[1]

4.3 Experimental design

The empirical study realized in this work was divided in two different experiments: Artificial and Participant-based. Basically, the first one is conducted with a simulator representing the DM, while the second one employs software engineering practitioners.

4.3.1 Artificial experiment

In order to represent the DM's role during the IGA interaction, a simulator was developed. The main purpose of this simulator is not to faithfully simulate a human being, but rather demonstrate the influence of a certain evaluation profile in the search process when faced with different scenarios of next release planning.

Similar to the used idea in Shackelford and Corne (2004) and Tonella et al. (2013), the human tacit assessment is simulated by creating a "target solution" which represents a solution that the DM would consider "ideal" or "gold standard". Such solution has the same structure of an individual and, consequently, comprises of a subset of the selected requirements to be implemented, respecting the budget and the interdependency constraints. The requirements present in the target solution are randomly chosen. In this experiment all instances were tested, where each one has the same specific target solution for the all 30 algorithm executions.

Throughout the interactions in the search process, the simulator provides a SE considering the target solution that is established for the respective instance. The evaluation given to a particular solution is proportional to how ideal it is, attending the implicit preference encompassed by the target solution. If the included requirements in a candidate individual are totally different from the target solution, the evaluation is minimal. On the other hand, when the individual is equal to the target solution, the evaluation is maximum. The evaluations are proportionally given for the other possibilities. Thus, the following equation is used by the simulator to determine a SE for a candidate solution:

$$simulatedSE(X, P) = \left(\frac{\sum_{j|1 \leq j \leq N \wedge x_j = p_j}^{N} 1}{N} \right) \times SE_{max}, \tag{15}$$

where, X is a candidate solution, P is the target solution, and N is the total number of requirements in the solution. Decision variable x_j indicates if the requirement r_j is included in the candidate solution, while p_j indicates whether the requirement r_j belongs to the target solution P. Finally, SE_{max} is the highest value SE can assume. For this empirical study, the minimum $simulatedSE(X, P)$ is 0 and the maximum is 100.

[1] http://goes.uece.br/allyssonaraujo/architecture4inrp.

4.3.2 Participant-based experiment

This experiment aims at verifying the behavior and feasibility of the architecture when it is used by software engineering practitioners. A group of 5 participants was invited to act as decision makers in the experiments. The members of the group have between 2 and 10 years of experience in the software engineering industry, adding up to a total of 30, and an average of 6 years of experience. Regarding the software development experience, on a scale of "low", "moderate"? and "high", four participants rated at "moderate" and one at "high". In terms of experience in requirements selection, four participants rated at "moderate" and one at "low". All participants have graduated in computer science or related areas, and have received or are finishing post-graduate degree.

Before the experiment, each participant was separately briefed about (i) the task they were supposed to perform, which was selecting requirements to be implemented in the next release, (ii) the architecture and interactive approach they would be using and (iii) the Word Processor instance they would be working with. More specifically, at first moment, the details of the next release problem were explained, including its motivation and major aspects such as multiple stakeholders, $score$ and budget constraint. After this step, the proposed approach was presented through the explanation of all components depicted in Fig. 1. In the final stage, initially a period of tool familiarization with a simple NRP instance was given and, finally, the Word Processor instance was presented. This real-world instance was chosen because it presents more details than ReleasePlanner, thus, being considered to be more intuitive. This explanation lasted between 35 and 40 minutes. Based on these details, a simple requirements specification document was produced and handled for all participants. This document consists of all requirement descriptions, the importance values given by the stakeholders and implementation efforts. Also, the total available budget and the relevance of each stakeholder to the company was presented. Due to the space constraint, just a sample piece of this document is reproduced in the Table 2.

Table 2 A sample piece of the requirements specification document shown to the participants

Budget #	850.20 Description	Weight Cost	9 Md. Fazlul Alam Chowdhury	3 Jim Li	5 Rick Bertuzzi	7 Samantha Holmes	Total
1	Create a new file	66.00	8	9	9	9	35
2	Open an existing file	76.00	8	9	9	9	35
3	Close current file	11.00	8	9	9	1	27
4	Save a file	63.00	8	9	9	9	35
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
50	Search a text in the document…	7.00	1	7	9	6	23

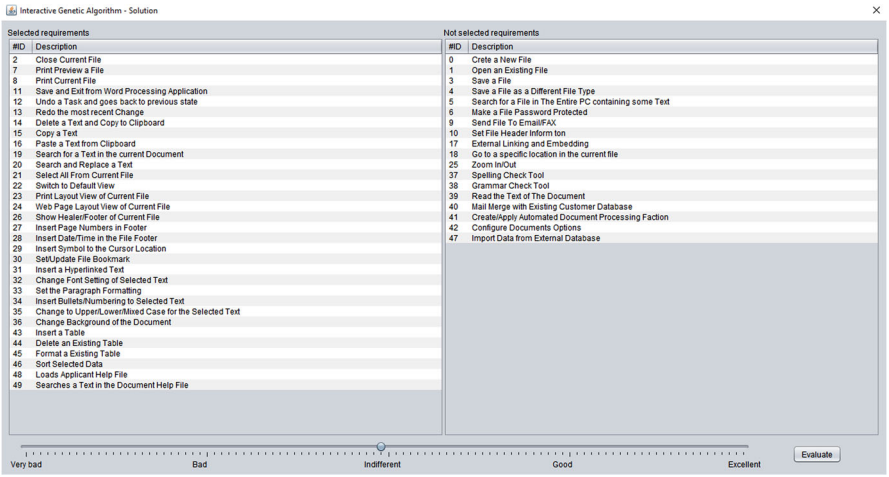

Fig. 4 Graphical user interface

Next, it was explained to each participant that he/she would need to perform the role of a requirements engineer in a hypothetical company in which the software to be developed is a Word Processor. Then, each participant would use the presented tool to select a set of requirements to be implemented in the next release, where the subjective evaluations could be based on the information detailed in the requirements specification document. As an attempt to assess the behavior of the approach when facing different evaluation profiles, participants had complete freedom to adopt any judgement criteria when giving subjective evaluations to each candidate solution.

Many of the issues faced when implementing an IEC technique can be resolved or avoided by careful design and evaluation of the existing experience and environment of the potential users. Good visualization is a key to the success of the IEC; therefore, significant effort should be put into the user interface design (Shackelford 2007). Thus, a graphical user interface (GUI) was developed to provide a better and user-friendly environment for the participants to interact with the implemented approach. As seen in Fig. 4, the requirements included in a solution to be evaluated are presented in the left-hand side, while the not selected requirements are displayed in the right-hand side. The participant gives the SE to the solution by adjusting the slide on the bottom of the screen. As it was previously explained, a set of solutions will be iteratively presented to the participant until the number of interactions is reached. In the end, the final solution is presented to the participant.

As stated earlier, before the search process, it is necessary to define the weights α and β in the fitness function, regarding the influence of the $score$ function and the SE value during the search process, respectively. It is also necessary to indicate how many interactions the DM will perform during the optimization process. Specifically, this empirical study used $\alpha = 1$ and $\beta = 1$ as weight configuration in the fitness function, and a total of 50 interactions for each participant was established.

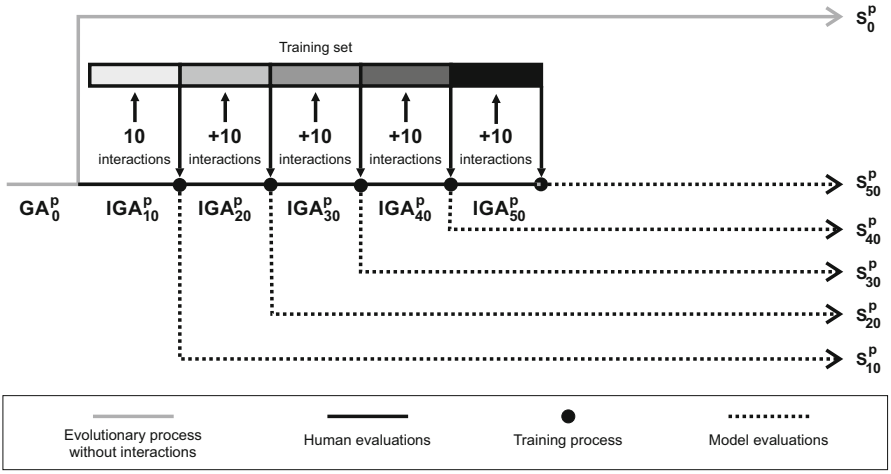

Fig. 5 Participant-based experiment procedure

In order to better exploit the experiments performed with the participants, a simple procedure was designed to evaluate the approach's performance when used with a different number of interactions, even though each participant interacted with the system only 50 times. Such a procedure is depicted in Fig. 5, and consists in training the learning model at different interactive cycles during the optimization process, consequently, replacing the DM at different moments. Consider IGA_i^p, where p represents the number of the participant and i the number of interactions using the interactive genetic algorithm. Similarly, S_i^p represents the solution generated by a participant p with i interactions. This way, the results with different number of interactions can be properly compared.

At first, a solution without any human influence is generated, called S_0^p. As can be seen, there are no interactions at this point; therefore, a standard Genetic Algorithm is used. This solution will be used to conduct comparisons between interactive and non-interactive methods, and also to provide the $score_{max}$ to be used in the $score$ normalization (Eq. 6). After this solution is captured, the first interactive cycle composed by the first 10 interactions is initiated. As previously presented, each interaction represents a candidate solution that receives a SE from the DM. Consequently, these 10 solutions and their respective subjective evaluations are included in the training dataset. Then, the remainder of the evolutionary process will be realized considering a learning model constructed with these 10 samples until a final solution S_{10}^p is found. After these first 10 interactions are reached, the same population continues to be used for a second interactive cycle of 10 interactions. At this point, the training dataset will be composed by the samples collected in the first one 10 interactions complemented by the 10 more samples captured in the second cycle. Naturally, the Learning Model will be constructed in this moment considering 20 samples to find a final solution S_{20}^p. This process is repeated until the 50 interactions are reached, in other words, the five cycles of 10 interactions is completed. In the end, 6 different solutions are found for each participant p, considering the different number of interactions i.

4.3.3 Research questions

Three research questions were designed to assess and analyze the behavior of the proposed approach. They are presented as follows:

- RQ_1 **(Sanity check)**: *Does the proposed architecture incorporate the subjective evaluations in the final solution?*
  In order to answer this question, there is analyzed if the final solutions are influenced by the interactions throughout the evolutionary process. The metric employed was the *Similarity Degree*, where percentually represents how similar a candidate solution is when compared to the target solution. Given the fact that is a exhaustive test which uses a huge number of interactions and weight configurations, just the Artificial Experiment was considered and, consequently, just the simulator.
- RQ_2 **(Interactivity trade-off)**: *What is the trade-off between the DM's satisfaction and the impact on* $score$ *values?*
  As explained in Sect. 3, the DM preferences towards the selected requirements may be not influenced by the $score$ value. Thus, it is natural to have some trade-off regarding the loss in the overall stakeholder's satisfaction to achieve a better solution from the DM subjective point of view. The metrics used to evaluate this aspect were the *PP* and *SF*. The first one indicates how much is lost in explicit information in order to incorporate implicit preferences, while the second metric shows the gain, in percentage terms, in *SD* by comparing a solution with DM influence and another one without DM influence. Such as previous research question, only the Artificial Experiment was considered.
- RQ_3 **(Comparison)**: *Does the proposed architecture improve the DM's satisfaction when compared to the non-interactive approach?*
  To answer this question, a participant-based experiment was conducted aiming to verify the feasibility of the architecture when it is used by professionals with different evaluation profiles. The metrics fitness (*F*), SE, *SF* and *PP* were used in the experiment. The *SD* it was not considered because there is not a previous target solution to represents the subjectively ideal solution to each participant, as well as simulated in the artificial experiment.

4.4 Results and analysis

The results for the empirical study are presented in this section using the analysis of the previous three presented research questions.

- RQ_1 **(Sanity check)**: *Does the proposed architecture incorporate the subjective evaluations in the final solution?*

The analysis conducted in this question aims at investigating the influence of the number of interactions in the evolutionary process and, consequently, the *Similarity Degree* (*SD*) achieved by the final solution. The higher *SD* value, more similar the candidate solution is to the target solution. Table 3 presents the average *SD* value for the 30 runs of the IGA with different number of interactions, considering all instances

Table 3 Average of *similarity degree* values for 30 runs and different number of interactions (i) using LMS_1 and MLP_2 with $\alpha = 1$ and $\beta = 1$

i	I_50					I_100					I_150				
	p	LMS_1	p	MLP_2	$\hat{A}_{12}$	p	LMS_1	p	MLP_2	$\hat{A}_{12}$	p	LMS_1	p	MLP_2	$\hat{A}_{12}$
10	–	62.00	–	65.47	**0.26**	–	55.23	–	57.57	**0.30**	-	58.62	–	58.64	0.48
20	△	63.00	▲	68.73	**0.22**	△	55.33	▲	61.40	**0.15**	△	58.73	△	60.29	0.35
30	▲	69.40	△	70.53	0.43	▲	58.67	△	63.80	**0.20**	△	60.07	▲	62.38	**0.33**
40	▲	74.20	△	70.73	**0.71**	▲	62.00	▲	67.40	**0.19**	▲	62.80	△	63.31	0.46
50	▲	79.60	△	72.07	**0.86**	▲	67.73	△	68.57	0.43	▲	64.78	▽	61.98	**0.69**
60	△	80.47	△	73.53	**0.86**	▲	71.03	▽	67.20	0.59	▲	67.67	△	64.93	0.64
70	▽	79.67	▲	76.67	**0.69**	▲	75.37	△	68.10	**0.78**	△	68.98	△	65.33	**0.66**
80	▽	79.60	▼	73.13	**0.85**	△	78.50	▲	71.80	**0.79**	▲	71.40	▽	64.51	**0.84**
90	△	80.47	△	74.80	**0.83**	▲	82.30	△	73.00	**0.90**	▲	75.09	△	66.38	**0.95**
100	△	81.93	△	75.53	**0.88**	△	83.47	△	74.57	**0.90**	△	75.96	△	67.04	**0.90**
200	▽	80.53	△	76.80	**0.72**	△	83.83	▽	73.83	**0.90**	▲	83.53	△	69.73	**0.99**
300	△	82.27	▽	76.00	**0.83**	△	84.17	▽	72.50	**0.94**	△	84.00	▽	69.53	**0.98**
400	△	83.07	△	76.40	**0.87**	▽	82.43	▽	70.33	**0.91**	▽	83.09	△	71.04	**0.92**
500	▽	81.73	▲	78.40	**0.71**	△	83.43	△	71.83	**0.97**	△	84.16	▽	68.09	**1.00**

Table 3 continued

i	I_200					Word Processor					ReleasePlanner				
	p	LMS_1	p	MLP_2	$\hat{A}_{12}$	p	LMS_1	p	MLP_2	$\hat{A}_{12}$	p	LMS_1	p	MLP_2	$\hat{A}_{12}$
10	–	55.08	–	56.30	0.39	–	49.33	–	56.67	**0.20**	–	70.93	–	76.53	**0.33**
20	▽	54.48	△	56.73	**0.29**	▲	56.53	▲	66.60	**0.18**	▲	77.33	▲	82.93	**0.33**
30	▲	56.72	▽	56.72	0.46	▲	60.80	▲	69.33	**0.22**	▲	90.00	△	83.73	**0.72**
40	△	58.25	▲	60.00	**0.35**	▲	75.33	△	69.80	**0.72**	△	91.33	△	84.53	**0.72**
50	▲	60.77	△	61.75	0.43	▲	82.93	△	71.80	**0.99**	△	94.53	▽	84.40	**0.85**
60	▽	60.50	▽	60.65	0.48	▽	81.87	▽	71.73	**0.96**	▽	90.80	▽	82.93	**0.75**
70	▲	63.63	▽	59.97	0.65	△	82.27	△	72.13	**0.94**	▽	88.80	△	84.67	**0.64**
80	▲	65.47	△	60.85	**0.71**	△	82.80	▽	70.33	**0.99**	△	90.40	▽	84.67	**0.64**
90	▲	68.10	△	62.50	**0.71**	▽	82.73	△	72.40	**0.96**	△	92.67	▽	84.40	**0.74**
100	△	69.98	▽	61.90	**0.82**	▽	82.40	△	73.00	**0.94**	▽	90.40	▽	84.27	**0.71**
200	▲	82.27	△	65.72	**0.98**	△	82.60	△	73.73	**0.97**	△	92.40	△	85.73	**0.72**
300	△	83.37	▽	64.85	**1.00**	△	82.93	▲	75.33	**0.96**	▽	90.67	△	89.07	0.56
400	▽	82.35	△	65.65	**0.99**	▽	82.07	△	77.40	**0.76**	▽	90.53	△	90.67	0.50
500	△	82.82	▽	65.47	**0.99**	△	82.13	▽	76.00	**0.81**	▽	87.07	△	93.60	**0.32**

The p column indicates statistical difference between *SD* values considering a 95 % confidence level, where the symbol △ means the average is higher but not significantly different, ▽ (lower but not significantly different), ▲ (significantly higher) and ▼ (significantly lower). The $\hat{A}_{12}$ column presents the *effect size* measure for the two machine learning techniques with the same number of interactions. Values in bold represent statistical difference

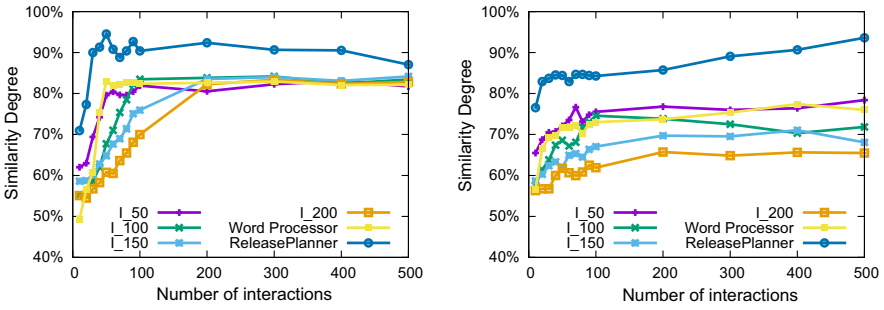

Fig. 6 Relation between *similarity degree* and number of interactions. **a** Machine learning technique: LMS. **b** machine learning technique: MLP

Table 4 Number of interactions for each instance and machine learning technique

	I_50	I_100	I_150	I_200	Word Processor	ReleasePlanner
LMS	60	100	200	300	50	40
MLP	200	200	200	200	200	200

and the two machine learning techniques. The sets of 30 values for each particular number of interactions are statistically compared in order to assess whether a different number of interactions yield a statistically different *SD* value.

When looking at the LMS results, it is possible to notice that, for every instance, there is no significant gain in *SD* after a certain number of interactions which, varies according to the instance size. For example, considering I_50, this stability is achieved after 50 interactions, while 200 interactions are needed for I_200. Such behavior can be visualized in Fig. 6(a), which shows the considerable *SD* increase in the first 200 interactions and, after a while, the mentioned stabilization.

Regarding the MLP results, the significant gains in *SD* can be verified and only occur using a smaller number of interactions. The non-significant differences for bigger numbers of interactions make it difficult to precisely identify the moment in which *SD* stabilizes. As presented in Fig. 6(b), such stabilization occurs at about 200 interactions for most of the instances. Furthermore in this analysis, LMS outperforms MLP in 80 % of the results. However, the MLP outperforms the LMS in 92 % when the number of interactions is <30.

The previously mentioned stabilization is closely associated with the machine learning technique and the size of the instance. Therefore, it is natural that the bigger the instance, the more difficult it will be to learn the implicit preferences. These stabilization values will be used in the next two research questions. For instance, for I_50 using LMS, the conducted analyses will be performed with a fixed number of 60 interactions because this is the number in which occurs the stabilization of the *SD* for this instance and machine learning technique. The number of interactions defined to all instances and machine learning techniques which will be further used are presented in Table 4.

In conclusion, it was observed the *SD* value has an intrinsic relation with the number of interactions. This can be explained because as the number of interactions increase, more samples (individuals and *SE* values) are included in the training dataset. Consequently, with a learning model properly adjusted, it is natural that the final solutions will be more similar to the target solution, i.e., more suitable in subjective aspects. In turn, these conclusions suggest the proposed approach passes the sanity check when it demonstrates that the implicit preferences given by the simulator are incorporated in the solutions.

– RQ_2 **(Interactivity trade-off)**: *What is the trade-off between the DM's satisfaction and the impact on score values?*

As previously discussed, it is natural to have some trade-off related to the loss in *score* to achieve a better solution in terms of the DM's subjective satisfaction. Thus, to provide an analysis of this trade-off, two complementary results will be detailed: the gain in *Similarity factor* (*SF*) and the loss in *Price of preference* (*PP*). Through the first metric it is possible to measure the proportional gain in *SD*, while the second helps to shed light in the expected impact in *score*.

First, the *SF* results will be analyzed while the β weight is increased in the fitness function. A higher *SF* value indicates a higher gain in *SD*. The experiments were performed in such a way that the α weight of the *score* function was fixed as 1, and the β weight of the SE function was varied from 0 to 1 with uniform intervals of 0.1. In other words, different scenarios are considered, ranging from no influence of the DM ($\alpha = 1$ and $\beta = 0$) to configurations in which the subjective evaluation is equivalent to the *score* function ($\alpha = 1$ and $\beta = 1$). Table 5 presents the average of *SF* values for 30 runs, considering different β weights and using both LMS and MLP.

For example, analyzing the real-word instance Word Processor in the configuration of $\beta = 0.5$, a *SF* value of 23.35 and 27.32 % was obtained for LMS and MLP, respectively. When the β weight was doubled to 1, the *SF* values had a considerable increase of 72.12 and 54.89 %, respectively. Generally, when comparing solutions without DM influence ($\alpha = 1$ and $\beta = 0$) and solutions influenced by him/her ($\alpha = 1$ and $\beta = 1$), the average gain of *SF* for all instances using LMS is 53.78 %, while using MLP is 34.06 %. The results for all instances are similar, which clearly demonstrates that *SF* values significantly grows as β increases.

Figure 7 shows the *SF* values obtained with LMS and MLP considering the proportional increase in β. These results reinforces the capability of the proposed approach at incorporating the subjective evaluations and, consequently, satisfy the DM's preferences. In addition, it shows that the influence of the DM's preferences in the search process can be adjusted by the weights configuration in the fitness function according to specific needs of different software projects.

As previously discussed, the incorporation of the DM subjective knowledge in the final solution usually accrues a loss in the *score* function. Therefore, an analysis of the *Price of Preference (PP)* is suitable. A higher *PP* value indicates a bigger loss in *score*. Similarly to the *SF* analysis, the experiments were performed in which the α weight of the *score* function was fixed in 1, and the β weight of the SE function varied from 0 to 1 with 0.1 intervals. Table 6 presents average *PP* values for 30 runs, considering different configurations of β weight and using LMS and MLP.

Table 5 Average of *SF* values for 30 runs and different settings of β using LMS_1 and MLP_2

β	I_50					I_100					I_150				
	p	LMS_1	p	MLP_2	$\hat{A}_{12}$	p	LMS_1	p	MLP_2	$\hat{A}_{12}$	p	LMS_1	p	MLP_2	$\hat{A}_{12}$
0.1	▲	5.27%	▲	10.54%	0.34	▲	8.80%	▲	10.30%	0.45	▲	5.62%	▲	5.57%	0.50
0.2	△	4.21%	▲	6.39%	0.40	▲	17.60%	▲	22.60%	**0.32**	▲	17.39%	▲	6.97%	**0.85**
0.3	▲	12.24%	▲	10.19%	0.58	▲	25.49%	▲	21.05%	0.63	▲	19.69%	▲	10.55%	**0.79**
0.4	▲	21.65%	▲	15.26%	**0.65**	▲	31.47%	▲	30.24%	0.54	▲	27.80%	▲	9.64%	**0.94**
0.5	▲	22.25%	▲	21.38%	0.54	▲	37.84%	▲	30.38%	**0.64**	▲	32.47%	▲	13.21%	**0.91**
0.6	▲	24.70%	▲	19.09%	0.63	▲	40.25%	▲	32.00%	**0.69**	▲	33.92%	▲	14.55%	**0.91**
0.7	▲	32.51%	▲	23.75%	**0.67**	▲	47.98%	▲	37.94%	**0.67**	▲	37.37%	▲	18.89%	**0.92**
0.8	▲	30.98%	▲	21.32%	**0.71**	▲	52.81%	▲	34.73%	**0.84**	▲	41.42%	▲	17.38%	**0.95**
0.9	▲	28.01%	▲	26.07%	0.57	▲	49.73%	▲	33.08%	**0.81**	▲	42.28%	▲	22.17%	**0.88**
1.0	▲	36.86%	▲	28.00%	**0.69**	▲	56.68%	▲	41.58%	**0.79**	▲	46.46%	▲	20.57%	**0.97**
β	I_200					Word Processor					ReleasePlanner				
	p	LMS_1	p	MLP_2	$\hat{A}_{12}$	p	LMS_1	p	MLP_2	$\hat{A}_{12}$	p	LMS_1	p	MLP_2	$\hat{A}_{12}$
0.1	▲	8.41%	▲	4.66%	**0.66**	△	1.51%	△	0.72%	0.55	▲	16.65%	▲	12.47%	0.46
0.2	▲	17.09%	▲	9.26%	**0.73**	▲	2.90%	▲	5.77%	0.43	▲	28.54%	▲	25.17%	0.48
0.3	▲	23.38%	▲	8.93%	**0.87**	▲	8.99%	▲	11.90%	0.39	▲	32.06%	▲	31.26%	0.43
0.4	▲	31.54%	▲	14.23%	**0.90**	▲	12.23%	▲	18.15%	0.36	▲	33.11%	▲	34.81%	0.29
0.5	▲	37.64%	▲	13.84%	**0.98**	▲	23.35%	▲	27.32%	0.36	▲	34.87%	▲	33.78%	0.41
0.6	▲	42.56%	▲	12.27%	**0.99**	▲	35.31%	▲	39.22%	0.46	▲	36.97%	▲	36.08%	0.34
0.7	▲	46.19%	▲	12.93%	**0.96**	▲	45.49%	▲	42.77%	0.54	▲	39.23%	▲	36.96%	0.45
0.8	▲	51.41%	▲	21.35%	**0.97**	▲	49.02%	▲	47.35%	0.51	▲	45.33%	▲	35.53%	**0.60**
0.9	▲	52.42%	▲	19.52%	**0.97**	▲	65.48%	▲	53.05%	**0.76**	▲	38.32%	▲	40.38%	0.40
1.0	▲	56.79%	▲	21.63%	**0.98**	▲	72.12%	▲	54.89%	**0.89**	▲	51.70%	▲	37.66%	**0.65**

The p column indicates statistical difference between *SF* values considering a 95 % confidence level, where the symbol △ means the average is higher but not significantly different, ▽ (lower but not significantly different), ▲ (significantly higher) and ▼ (significantly lower). The $\hat{A}_{12}$ column presents *effect size* measure for the two machine learning techniques with the same β configuration. Values in bold represent statistical differences

Analyzing the instance I_150 in the configuration of $\beta = 0.5$, the *PP* loss was 2.67 and 1.01 % for LMS and MLP, respectively. Doubling this weight to $\beta = 1$, the results are 6.00 and 2.26 % for the LMS and MLP, respectively. In summary, the average *PP* loss for all instances using LMS was 10.62 %, while employing MLP was 6.22 %. These values are obtained when comparing solutions generated without any DM influence ($\alpha = 1$ and $\beta = 0$) with solutions generated using the same weight for functions $score$ and SE ($\alpha = 1$ and $\beta = 1$).

Thus, it is important to highlight that the loss in *PP* using MLP being smaller than in the LMS one, and is a direct consequence of the reached *SF* in the previous analysis. In other words, as LMS reaches a higher *SF* and can incorporate most of the DM

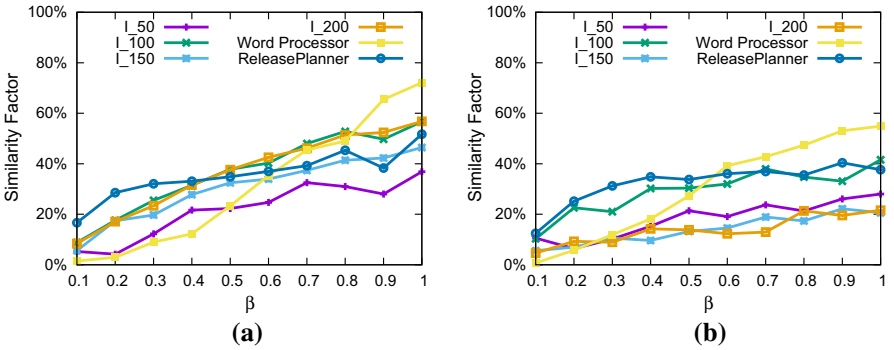

Fig. 7 Relation between the increase of the β weight and *similarity factor*. **a** machine learning technique: LMS. **b** machine learning technique: MLP

preferences, the *PP* loss is naturally proportional. Similarly to the *SF* values, the *PP* values also significantly grow in relation with the β increase in the most of instances.

Figure 8 presents an overview of the *PP* behavior with LMS and MLP when the β weight of SE in the fitness function is incremented. As can be seen, the solutions found for almost all instances present a similar increasing tendency of *PP* values for both machine learning techniques. Thus, the higher is the influence of the implicit preferences in the search process, the smaller will be the $score$ function. Such as in the *SF* analysis, in the scenario where the loss in $score$ needs to be avoided, another weight configuration that is more suitable for the specific software project can be employed. However, even in the case in which the weights are balanced ($\alpha = 1$ and $\beta = 1$), the *PP* loss is, in average, merely 10.62 and 6.22 % for the LMS and MLP, respectively. This *PP* loss can be considered small, since there is a substantial gain in *SF* of 72.12 and 54.89 % for both machine learning techniques.

From the previous analyses, two important findings were obtained: (i) the capability of the proposal at incorporating the subjective evaluations and, consequently, including the implicit preferences; and (ii) how much is lost in terms of $score$ due to the inclusion of subjective evaluations.

In order to provide a better visualization of the trade-off between the metrics *SD* and *PP*, the Fig. 9 presents their relation considering all instances and both machine learning techniques. The right-hand side of the figures shows how much is reached of *SD*, while the left-hand side reflects how much is lost in $score$ for all instances. For example, looking at instance I_100 and using the LMS, one needs to lose only 7.0 % of $score$ in order to reach a *SD* of 82.3 %. Using MLP the $score$ loss was 5.0 % in order to reach a 73.8 % of *SD*.

In synthesis, when using LMS, the average loss of $score$ for all instances is 10.62 % and the average gain in *SD* is 83.99 %. By adopting MLP, the average values were 6.22 and 74.26 % for the lost of $score$ and gain in *SD*, respectively. Therefore, it can be concluded that the proposed architecture is capable to incorporate the DM's preferences in the iNRP, with a considerably small loss in score.

– RQ_3 **(Comparison)**: *Does the proposed architecture improve the DM's satisfaction when compared to the non-interactive approach?*

Table 6 Average of *PP* for 30 runs and different configuration of the β weight using LMS_1 and MLP_2 with $\alpha = 1$

β	I_50					I_100					I_150				
	p	LMS_1	p	MLP_2	$\hat{A}_{12}$	p	LMS_1	p	MLP_2	$\hat{A}_{12}$	p	LMS_1	p	MLP_2	$\hat{A}_{12}$
0.1	▼	−0.72%	△	0.91%	**0.34**	▽	−0.24%	△	0.33%	0.45	▽	−0.30%	△	0.11%	0.46
0.2	▽	−0.02%	△	0.43%	0.41	▲	0.85%	▽	−0.31%	0.59	▽	−1.00%	▽	−0.12%	0.43
0.3	▲	0.87%	▲	0.83%	0.54	▲	1.79%	▲	1.76%	0.53	△	0.96%	△	0.24%	0.57
0.4	▲	3.43%	▲	1.30%	0.72	▲	1.29%	▲	1.58%	0.5	△	0.25%	△	1.24%	0.40
0.5	▲	4.24%	▲	2.51%	0.63	▲	3.71%	▲	1.48%	**0.8**	▲	2.67%	▲	1.01%	0.64
0.6	▲	4.87%	▲	3.27%	**0.73**	▲	4.91%	▲	2.94%	**0.72**	▲	3.13%	△	0.47%	**0.74**
0.7	▲	6.34%	▲	4.39%	**0.68**	▲	5.77%	▲	3.85%	**0.75**	▲	3.68%	△	0.35%	**0.84**
0.8	▲	7.54%	▲	4.83%	**0.72**	▲	7.69%	▲	3.06%	**0.90**	▲	4.69%	▲	1.73%	**0.76**
0.9	▲	8.01%	▲	6.24%	0.66	▲	7.76%	▲	3.48%	**0.86**	▲	5.64%	△	1.07%	**0.86**
1.0	▲	8.86%	▲	6.46%	**0.72**	▲	8.71%	▲	5.03%	**0.82**	▲	6.00%	▲	2.62%	**0.88**
β	I_200					Word Processor					ReleasePlanner				
	p	LMS_1	p	MLP_2	$\hat{A}_{12}$	p	LMS_1	p	MLP_2	$\hat{A}_{12}$	p	LMS_1	p	MLP_2	$\hat{A}_{12}$
0.1	▲	0.25%	△	0.20%	0.53	▲	0.69%	▽	−0.08%	0.63	▲	0.07%	▼	−0.44%	0.57
0.2	△	0.14%	△	0.45%	0.46	△	0.17%	△	0.23%	0.50	▲	0.82%	▲	0.95%	0.52
0.3	▲	1.05%	△	1.15%	0.49	△	0.90%	△	0.74%	0.44	▲	1.20%	▲	1.83%	0.38
0.4	▲	2.08%	▲	0.88%	**0.64**	▲	1.90%	▲	2.67%	0.38	▲	1.59%	▲	1.46%	0.48
0.5	▲	4.36%	△	1.11%	**0.84**	▲	4.80%	▲	4.99%	0.44	▲	2.89%	▲	2.00%	0.55
0.6	▲	5.15%	▲	1.74%	**0.89**	▲	8.65%	▲	9.04%	0.52	▲	2.99%	▲	2.75%	0.49
0.7	▲	5.65%	▲	2.10%	**0.85**	▲	12.22%	▲	10.11%	0.67	▲	3.58%	▲	2.67%	0.48
0.8	▲	6.87%	▲	3.17%	**0.90**	▲	13.08%	▲	12.69%	0.45	▲	6.67%	▲	2.61%	**0.60**
0.9	▲	7.59%	▲	2.85%	**0.93**	▲	20.15%	▲	15.54%	**0.76**	▲	4.98%	▲	3.61%	0.48
1.0	▲	8.12%	▲	3.87%	**0.90**	▲	24.63%	▲	15.95%	**0.97**	▲	7.43%	▲	3.42%	**0.66**

The p column indicates statistical difference between *PP* values considering a 95 % confidence level, where the symbol △ means the average is higher but not significantly different, ▽ (lower but not significantly different), ▲ (significantly higher) and ▼ (significantly lower). The $\hat{A}_{12}$ column presents *effect size* measure for the two machine learning techniques with the same β configuration. Values in bold represent statistical differences

As explained in Sect. 4.3.2, a group of software engineering practitioners was invited to use the proposed approach in a Participant-based Experiment. The obtained results for all participants are reported in Table 7.

As can be seen, there are no *SF* and *PP* values for the S_0^P configuration. This is natural because it is a non-interactive solution. Nevertheless, its values in Fitness and SE were calculated. The first one was measured by normalizing the value of the $score$ function to 100 and adding the *SE* value.

In terms of solution quality, analyzing specifically the Participant 4, the S_{50}^P with MLP was the best one among all interactive solutions (including other participants). Comparing its fitness value with the result of S_0^P, it is verified an improvement of

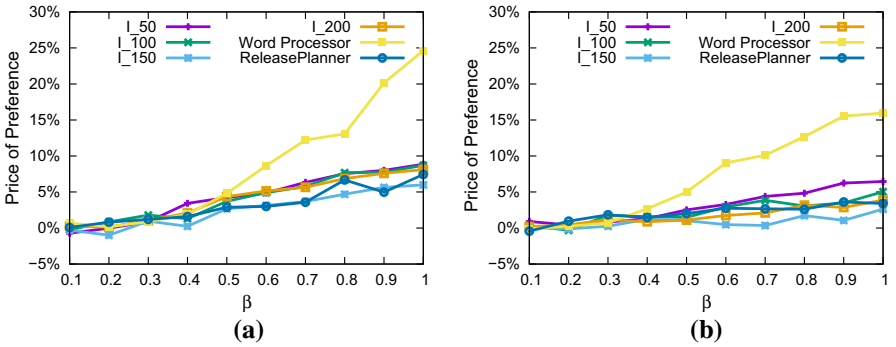

Fig. 8 Relation between *price of preference* and the increase of the β weight. **a** machine learning technique: LMS. **b** machine learning technique: MLP

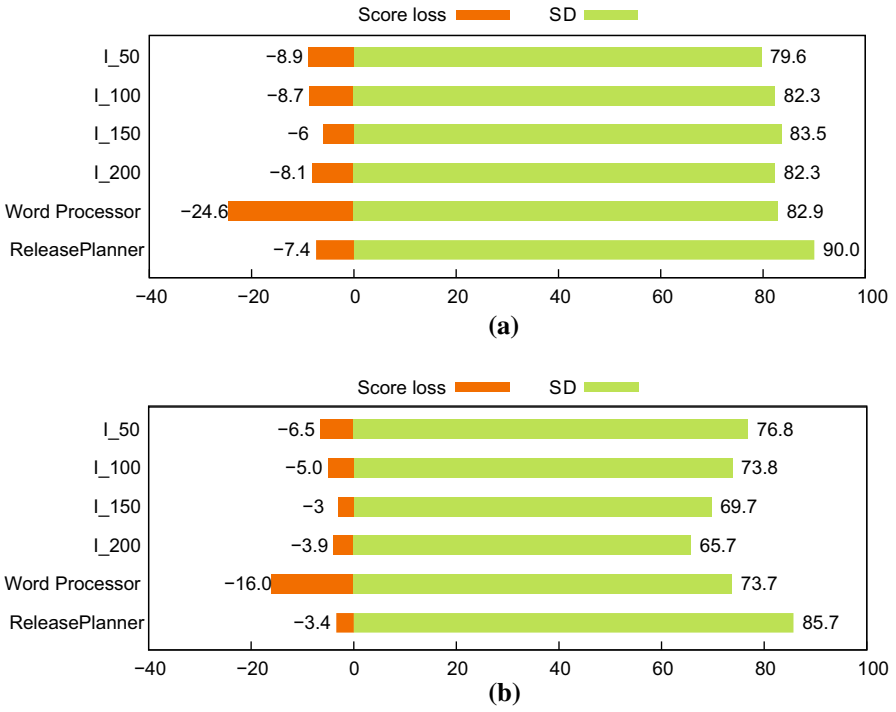

Fig. 9 Relation between *similarity degree* and *price of preference*. **a** machine learning technique: LMS. **b** machine learning technique: MLP

36.9 % and a *score* loss of 4.4 % (*PP* line). On the other hand, 94.17 was the fitness value of the worst solution, which is the S_{20}^P with LMS. The quality of this solution is 32.7 % worse than S_0^P. A discussion about the performance of LMS and MLP on the Participant-based Experiment will be presented later in Sect. 4.5.

Considering the fitness (*F* line) average results, the values of MLP from S_{20}^P to S_{50}^P, and, the values of LMS for S_{40}^P and S_{50}^P were higher than the fitness average obtained

Table 7 Values of fitness (F), SE, SF and PP for all participants with different number of interactions and each machine learning technique

Participant (p)		S_0^p	S_{10}^p		S_{20}^p		S_{30}^p		S_{40}^p		S_{50}^p	
			LMS_1	MLP_2	LMS_1	MLP_2	LMS_1	MLP_2	LMS_1	MLP_2	LMS_1	MLP_2
1	F	140.00	96.95	117.71	94.17	174.27	107.22	184.12	108.15	183.70	173.67	191.69
	SE	40.00	25.00	19.00	57.00	62.00	65.00	56.00	85.00	85.00	34.00	95.00
	SF	–	−37.50	−52.50	42.50	55.00	62.50	40.00	85.00	112.50	−15.00	137.50
	PP	–	0.10	0.84	3.00	8.95	3.16	13.00	6.188	12.860	23.20	4.41
2	F	162.00	165.68	189.00	173.04	190.01	186.00	184.90	174.94	173.38	176.53	182.11
	SE	62.00	62.00	61.00	75.00	75.00	73.00	89.00	75.00	87.00	75.00	85.00
	SF	–	0.00	−1.61	20.97	20.97	17.74	43.55	20.97	40.32	20.97	37.10
	PP	–	7.58	10.77	7.62	10.07	7.44	15.104	27.538	26.622	23.465	17.89
3	F	161.00	142.69	187.14	141.40	156.18	185.03	173.85	170.49	186.94	171.85	189.18
	SE	61.00	88.00	100.00	91.00	71.00	56.00	93.00	26.00	76.00	51.00	45.00
	SF	–	44.26	63.93	49.18	16.39	−8.20	52.46	−57.38	24.59	−16.39	−26.23
	PP	–	13.51	9.88	5.81	7.40	12.05	7.62	26.61	10.08	25.20	7.76
4	F	200.00	124.51	185.89	187.43	187.72	121.05	188.29	174.10	190.41	178.27	180.81
	SE	100.00	82.00	100.00	100.00	100.00	100.00	100.00	100.00	82.00	91.00	82.00
	SF	–	−18.00	0.00	0.00	0.00	0.00	0.00	0.00	−18.00	−9.00	−18.00
	PP	–	7.87	11.42	9.84	10.12	8.63	9.54	24.08	10.72	19.61	20.21
5	F	173.00	119.55	136.41	100.25	150.22	150.22	181.35	143.93	162.75	186.06	177.39
	SE	73.00	76.00	76.00	63.00	78.00	74.00	84.00	75.00	93.00	78.00	82.00

Table 7 continued

Participant (p)		S_0^p	S_{10}^p		S_{20}^p		S_{30}^p		S_{40}^p		S_{50}^p	
			LMS_1	MLP_2	LMS_1	MLP_2	LMS_1	MLP_2	LMS_1	MLP_2	LMS_1	MLP_2
	SF	–	4.11	4.11	−13.70	6.85	4.11	15.07	2.74	27.40	6.85	12.33
	PP	–	12.86	8.82	34.11	13.81	15.29	2.65	14.58	14.79	13.48	19.88
Average	F	147.20	115.48	143.49	131.82	149.23	141.46	156.88	149.69	159.70	149.34	164.90
	SE	74.00	66.60	71.20	77.20	77.20	73.60	84.40	72.20	84.60	65.80	77.80
	SF	–	−1.43	2.79	19.79	19.84	15.23	30.22	10.27	37.36	−2.52	28.54
	PP	–	8.38	8.35	12.08	10.07	9.32	9.58	19.80	15.02	20.99	14.03

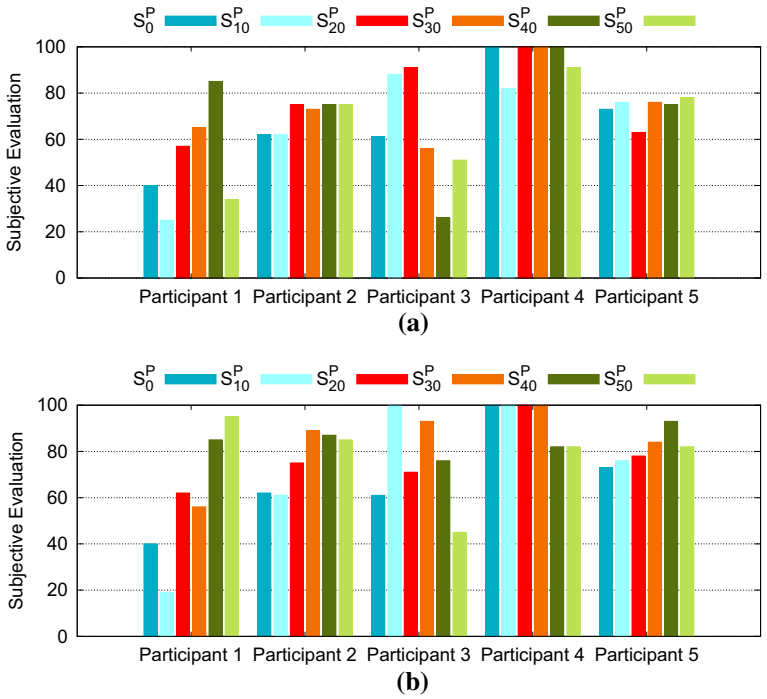

Fig. 10 Results of SE for each participant with several number of interactions. **a** machine learning technique: LMS. **b** machine learning technique: MLP

by S_0^P. Thus, in spite of the loss in *score* denoted by *PP*, an overall improvement in solution quality by the inclusion of DM interactions was observed.

Looking at SE and *PP* average of the results of S_{50}^P with MLP technique, it is noticed the values of both metrics were similar to the results obtained on Artificial Experiment (Tables 3 and 6) considering the same instance (Word Processor) with 50 interactions. It is important to remember that a simulator was used in Artificial Experiment, then *SD* metric in these tables is compared with *SE* value. This observation points out to the consistence of proposed approach when used by software engineering practitioners in comparison with the main conclusions achieved in the previous experiment.

In order to summarize the participants behavior throughout the experiment, Fig. 10a presents the SE given by the participant to the final solutions, as well as the number of required interactions.

Analyzing the Fig. 10a, which presents the LMS results, it is noticed that for all participants, there were at least two solutions that are better or at least as good as S_0^p (solution generated without interactions). Looking at the results of Participant 2, all interactive solutions are better than S_0^p, with an exception of S_{10}^p that had the same value of S_0^p.

Similarly, Fig. 10b presents the obtained values using the MLP technique. It is possible to realize that, for each participant, at least four interactive solutions present higher SE than the S_0^p solution, with exception of Participant 4, for whom three solu-

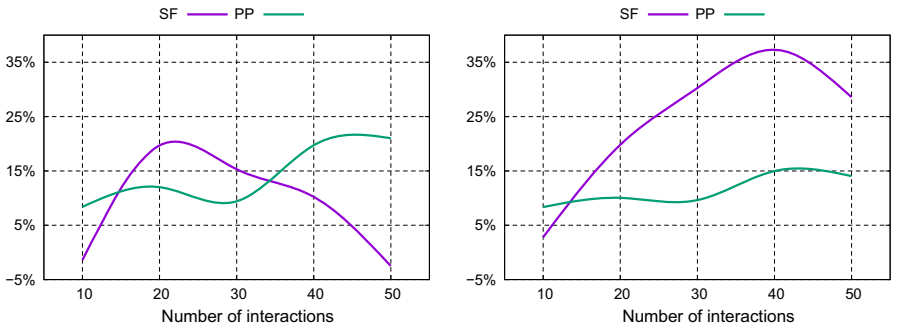

Fig. 11 Relation between *similarity factor* and *price of preference* considering the average of all participants for each number of interactions. **a** machine learning technique: LMS. **b** machine learning technique: MLP

tions were equal to S_0^p. In the specific case of Participant 5, all solutions influenced by him/her were superior than the non-interactive one.

Some interesting results to be highlighted are the ones obtained by the Participant 4. He/she provided a maximum subjective evaluation for S_0^p, thus, it was impossible for any other solutions to be better in terms of SE. This should not be a problem in a real usage of the proposed approach, because it is expected that if the non interactive solution already satisfies all the DM's desires, he/she would simply accept it.

Besides SE results, the trade-off between satisfaction of the DM's preferences and the $score$ loss is relevant in this experiment. Thus, Fig. 11a and b show the SF and PP averages of the solutions obtained by all participants with each number of interactions.

Specifically analyzing the results obtained with 50 interactions, LMS reached around 20 and 0 % of PP and SF, respectively, while MLP achieved almost 30 % of SF and <15 % of PP. More generally, in average, the SF values using LMS were higher than the PP ones in two of the five different number of interactions. Furthermore, SF values from MLP were bigger than PP values in four cases.

4.5 Discussion

Based on the data collected and analyzed in the empirical study, some interesting findings can be discussed. The first one is the relationship between the achieved results and the machine learning technique employed. In the Artificial Experiment (Sect. 4.3.1), a simulator was used to evaluate the IGA candidate solutions. The criteria adopted by the simulator to evaluate the solutions is always the same, in other words, the subjective evaluations are directly related to the number of requirements present in the candidate solution that are also present in the target solution. If the requirements included in a candidate individual are totally different from the target solution, the evaluation is minimal. Similarly, when the individual is equal to the target solution, the evaluation is maximum. As can be seen, the subjective evaluations provided by the simulator follow a linear proportion. On the other hand, the participant-based experiment (Sect. 4.3.2) employs real human evaluations which are defined as multi-criteria and of nonlinear character (Piegat and Sałabun 2012).

Hence, since the LMS is a robust regression linear model (Rousseeuw 1984), it is expected that it will reach good results when considering the simulator in the artificial experiment. Linear regression is a simple method that has been widely used in statistical applications. However, according to Witten and Frank (2005), there are two serious disadvantages to median-based regression techniques: (i) it can only represent linear boundaries between classes, which makes them too simple for many practical applications and (ii) they incur high computational cost, which also makes them infeasible for practical problems.

In contrast, the MLP technique, powered by the backpropagation training rule, is able to perform nonlinear regression; therefore, overcoming the LMS weaknesses. If the training examples are not linearly separable, the training rule converges toward a best-fit approximation to the target concept (Witten and Frank 2005; Mitchell 1997). As stated in Palit and Popovic (2006), the practical use of neural networks has been recognized mainly because of distinguished features such as (i) general nonlinear mapping between a subset of the past time series values and the future time series values and (ii) capability of learning and generalization from examples using a data-driven self-adaptive approach. These aspects theoretically underlie the conclusions of the empirical study suggesting why the LMS achieved better results in the Artificial Experiment, while the MLP outperforms the LMS in the Participant-based Experiment.

Another aspect that can be highlighted are the insights obtained into how people go about using the system. Firtly, all five participants were invited to sign a term of consent and to respond a questionnaire. On a scale of "insufficient", "indifferent" and "sufficient", all the participants considered the briefing (described in Sect. 4.3.2) before the experiment as "sufficient". When asked to rate how effective they found the experience of interactively selecting the requirements for the next release, all professionals considered as "effective". The scale used to this answer was "very ineffective", "ineffective", "indifferent", "effective" and "very effective". Complementing and reinforcing this achievement, it was asked if they would use the proposed approach in their workplaces. Following a scale of 1 ("no way) to 5 ("certainly"), one participant rated at 3, three participants at 4 and one at 5. These results may suggest two conclusions: (i) the proposed approach was considered distinguished by software engineering practitioners and (ii) it encourages the application of the presented approach in a real-world scenario of requirements planning.

The participants were also asked to rate how much tiring was to evaluate the candidate solutions during the IGA evolution. On a scale of 0 to 9, one participant rated at 4, two at 5 and two at 7, which generates an average of 5.6. As stated in Wang et al. (2006), "it is pretty hard to measure the fatigue reduction in the case of IEC, since IEC deals with subjective evaluation values that depend on the application task and the subject's perceived value of the task". There are some pieces of work that propose approaches to reduce the human effort without compromising the results and consistencies, such as Kamalian et al. (2006), Hsu and Huang (2005) and Wang et al. (2006). Even though these papers claim to reduce the effort undertaken, but the analysis of "how much" fatigue was mitigated is very context-depending. A straightforward way to assess some kind of improvement in human fatigue through the usage of the learning model would be the clock time taken to complete the task. However, this would not be the most appropriate way to analyze the human fatigue. It is reasonable to believe

that a certain participant may spend more time than another and still report less fatigue or, in other case, evaluate less solutions than other participant and indicates a higher fatigue. Also, it is possible that the participant could undertake the experiment with some previous tiredness (physical or psychological) which can have a direct influence in his/her fatigue report. Definitely, this is an important aspect to be properly discussed and refined.

Regarding the "free text" comments participants were allowed to give about the experience, it can be highlighted the need to integrate detailed requirements specifications in the user graphic interface. Others suggestions to the interface were proposed, such as increase the slide button and a more intuitive use of colors.

Finally, the answers of each research question discussed in the empirical study can be formalized as:

- RQ_1: *the solutions provided by the approach are properly influenced by the interactions throughout the evolutionary process, given the higher values obtained in Similarity Degree. This influence produces solutions that incorporate the subjective evaluations and, consequently, are more suitable to the Decision Maker preferences.*
- RQ_2: *the loss in score in the solutions is considerably small, especially when taking into account the high gain in Similarity Factor. This conclusion suggests that the approach is able to generate solutions as good as those produced without any human influence, but satisfying most of the Decision Maker preferences. In addition, this trade-off can be conveniently calibrated according to the software projects specific needs through a detailed weights configuration.*
- RQ_3: *even considering participants with different and unknown evaluation profiles, the proposed architecture was able to improve their subjective satisfaction. This fact can be justified through the higher* SE *values obtained in the Participant-based Experiment. Furthermore, the found results shown architecture consistence regarding to the conclusions achieved in the Artificial Experiment.*

4.6 Threats to validity

A list of threats to the validity of empirical studies in Search Based Software Engineering is presented in de Oliveira Barros and Neto (2011). Such threats are classified as: (i) conclusion validity threats, (ii) internal validity threats, (iii) construct validity threats and (iv) external validity threats. The above the threats that may effect the validity of the experiments performed in this paper will be discussed, including what was eventually designed to mitigate the effects of each one.

Regarding the *conclusion validity threats*, due to the stochastic nature of the IGA, each execution can achieve different results. Therefore, in order to ensure a fair comparison for each weight configuration, number of interactions, instances and machine learning technique, 30 runs were performed. According to Arcuri and Briand (2014), this is an acceptable number of executions for empirical studies in SBSE, even though a greater number of runs would have provided greater accuracy. Aiming at precisely providing conclusions regarding the achieved results, statistical analyses were performed using the Mann–Whitney U test to measure the difference between the samples with

a 95 % confidence level and, complementing this one, the Vargha and Delaney's $\hat{A}_{12}$ to highlight the *effect size*. Despite being extensively explored problem in the literature, there is no other approach using an IGA to solve the NRP. Given the lack of a recognized benchmark example, this work is focused in comparing the results reached by the proposed architecture with the ones generated without human influence, which can be considered a small scale investigation. Another important challenge to be discussed concerns to how evaluate and estimate the reduction of fatigue. This work asked the participants an numerical value which can express his/her tiredness during the experiment. Another more refined metric may provide more useful insights.

Concerning the *internal validity threats*, the parametrization procedure adopted to the empirical study is properly presented in the Sects. 4.2.2 and 4.2.3 , which present all the configurations regarding both the machine learning technique and IGA. With an aim to have a more generic analysis, the LMS and MLP were chosen due to the different learning strategies of each one. These main differences were discussed in their respective topics. Although the results were obtained from a preliminary empirical study, the tuning parametrization process was not conducted as suggested in Arcuri and Fraser (2011), which may indicate better configurations and, consequently, to produce superior results for some specific instance. Throughout the paper, there are no details regarding the code instrumentation, however, to mitigate this threat and facilitate the experiments replication, the source code and all evaluated instances can be found in the supporting web page.[2] Regarding the data collection procedure, all details of the design and generation of artificial instances, as well as the real-world instances particularities, are described in the Sect. 4.2.1. Furthermore, Participant-based Experiment are highly dependent of the developed scenario and, consequently, the users engagement. In order to mitigate this problem, before the experiment, each participant was separately briefed about the main details related to the experiment. Nevertheless, providing to the participants a more elaborate scenario thats is as similar as possible to their workplaces is a considerable challenge.

In relation to the *construct validity threats*, this paper does not perform a study on computational cost of execution or performance of the used algorithm. Aiming to be more meaningful and appropriate to the empirical study, it was necessary to develop a suite of metrics for the interactive perspective. All these metrics were precisely defined and can be reused in other works involving interactive optimization in SBSE. However, a specific metric for the learning perspective would be useful to consider. The proposed iNRP formulation is a generalization based on the mathematical model proposed in Baker et al. (2006), which can be considered a simplification of a real next release planning situation. Another issue to be discussed is the choice of the target solution that determines the performance of the *SD* and *SF* metrics being compared. As explained in Sect. 4.3.1, the requirements present in the target solution are randomly chosen, but of course there is no warranty that this is indeed the optimal selection. Regarding the type of the SE, a numerical value range was used, but other ways of evaluation may be show as more intuitive (e.g., "bad" to "excellent" or "one star" to "five star" rating).

[2] http://goes.uece.br/allyssonaraujo/architecture4inrp.

Finally, concerning *external validity threats*, artificial instances with 25, 50, 100, 150 and 200 requirements and real-word instances with 25 and 50 requirements were used in the empirical study. Experiments with bigger instances would have provided more generalizable results. Moreover, the number of professionals in the Participant-based Experiment could be bigger to provide a more reliable evaluation of the approach. To cope with this threat, participants with reasonable software engineering practice were invited to discuss the system usage.

5 Related work

In this section, the work related to this research is discussed. Firstly, the work directly related to the Next Release Problem, followed by the work that applies interactive optimization in Search Based Software Engineering. Finally, some strategies designed to handle human fatigue are also presented.

5.1 Next release problem

The first single objective formulation of the NRP was presented in Bagnall et al. (2001). This approach considers the existence of more than one customer with different levels of relevance to the company, which is indicated by his/her respective weight. Each customer then indicates which requirements he/she wants to be implemented in the next release. The customer is considered satisfied if all his/her requirements are selected for the next release. Also, each requirement presents a development cost. Thus, the goal is to select a set of requirements that maximizes the sum of satisfied customers' weights, while the implementation cost of the release is subject to the available release budget. Regarding the empirical evaluation, it uses exact methods, neighborhood heuristics and a metaheuristic called Simulated Annealing. It was noticed that the exact method has found better solutions for smaller instances in viable times, but in bigger instances the metaheuristic had better performance.

Differently from Bagnall et al. (2001), the work Baker et al. (2006) argues that each requirement should have its importance given by the customers. The global importance of a requirement is calculated by a weighted sum of the specific importance given by the customers. Thus, it aims at selecting a set of requirements that maximizes the global importance of the release. Such work is considered relevant due to the usage of real-world data from a great telecommunication company. In order to validate the approach, Greedy Algorithms and Simulated Annealing were applied, with the results being compared to the ones produced by an expert. It was concluded that both algorithms had a better performance than the professional.

Naturally, the NRP was tackled by various approaches. In Jiang et al. (2010), the usage of a hybrid algorithm composed by Ant System and Hill Climbing is proposed. The work in del Sagrado et al. (2010) is related to the use of the Ant Colony Optimization to the NRP, and the achieved results are better in solution quality and convergence terms than the results found by Genetic Algorithms and Simulated Annealing. Such work was complemented in do Nascimento Ferreira and de Souza (2012), when it was elaborated a comparative study between Ant Colony Optimization, Genetic Algo-

rithm and Simulated Annealing considering requirements interdependencies. This work concludes that the Ant Colony Optimization reaches better results than other metaheuristics.

In van den Akker et al. (2005), techniques of integer linear programming were applied considering some practical aspects of the NRP, such as the list of requirements, interactions between requirements and their respective costs, and the resources that the development team requires. The work on Xuan et al. (2012) focuses on solving big instances of the NRP through an algorithm named "Backbone Algorithm". The achieved results are compared with a Simulated Annealing variant known as LMSA.

Generally speaking, the approaches presented above can be considered decision-making tools, where the DM inserts data, then the tool automates the process and returns a set of requirements to be implemented in the next release. Due to this automatization, such approaches do not consider implicit preferences during the search process, and consequently don't making use of various benefits that human knowledge could offer to improve the results.

5.2 Interactive optimization in search-based software engineering

The work in Pitangueira et al. (2015) presents a systematic review and mapping study of SBSE approaches to requirements selection and prioritization. The authors point out that, in terms of innovation for the area, adding user judgement in the model may lead to better results, and would also take the empirical studies closer to the software engineering reality. This point of view is also reinforced in Zhang et al. (2008) and Harman et al. (2012).

In the requirements engineering field there are four pieces of work that can be highlighted. First of all, the first version of this work presented in Araújo et al. (2014) primarily assess whether an IGA for the NRP can properly incorporate the DM knowledge in the final solutions. In addition, it has also drafted a first version of the architecture that considers a learning model as an option of replacing the DM when necessary.

A conceptual proposal is presented in Pitangueira (2015), which focuses on improving the selection of software requirements for a next release. It proposes the usage of a search-based approach interactively with multiple stakeholders in the optimization process through the establishment of a judgment consensus about what is a good Pareto Front.

In Dantas et al. (2015), an interactive approach to the software release planning is proposed in which the search is guided according to a Preferences Base that is interactively supplied by the DM during the search process. In this approach, the fitness of a certain candidate solution is penalized according to the importance level of each preference that was not satisfied. Preliminary results indicates that the solutions are able to satisfy almost all user preferences, prioritizing the most important ones, with little $score$ loss.

Finally, Tonella et al. (2010) investigates an IGA approach for a real case study in the requirements prioritization process. The main idea of this paper is to minimize the amount of "requirements peers" evaluations obtained from the users, making this

approach more scalable and accurate concerning the final requirements prioritization. The results are positive denoting that the performance of the requirements prioritization process is better when employing the interactive approach. Later, this approach was extended in Tonella et al. (2013), pointing out comparisons with the IAHP, the state of the art reference algorithm for interactive requirement prioritization.

Regarding software maintenance, an approach to find appropriate refactoring suggestions using a set of examples is proposed in Ghannem et al. (2013). The fitness function combines the structural similarity between a candidate design model and refactoring examples, alongside developer's ratings for the refactorings proposed during execution of the IGA. The fitness function of the solution is computed as an average of its old fitness function and the overall designer's rating. Initially, the base of examples and an initial model to be improved are used as inputs. Results showed that this approach is stable regarding its accuracy, integrity, type and amount of refactorings per class.

Under the context of software design optimization, an inclusion of the developer in tasks of software re-modularization is proposed in Bavota et al. (2012). In such work, quality and dependencies between modules are automatically evaluated. The user, in turn, evaluates if two components must be in the same module or not. Given this feedback, a penalty function is applied to penalize solutions that violate the constraints imposed by the developers. Despite the effectiveness regarding cohesion, the approach does not consider the developer's knowledge when deciding about grouping the components.

Still considering the software design field, there is a large amount of work applying interactive optimization concepts (Parmee et al. 2006; Simons and Parmee 2010; Simons et al. 2010; Simons 2011; Simons and Parmee 2012; Simons et al. 2014; Simons and Smith 2013). It is possible to highlight the work in Simons et al. (2014) because it summarizes many ideas and concepts presented in the previous ones. The research is focused on interactive ant colony optimization in which the search process is guided by an adaptative model that bring together objective and subjective factors. Regarding the interaction, the user is invited to give a numeric evaluation (from 1 to 100) for a feasible candidate solution. The solution's representation is modelled as a UML diagram in which each class is divided into three compartments and connected with other classes through arrows. Concerning the results, the participants of the experiments rated the proposal as persuasive in the sense that it may be considered as an interesting direction in the interactive search for problems related to software design.

With respect to software testing, it is possible to cite the project exploited in Marculescu et al. (2012, 2013, 2015, ?). Generally, it is proposed a system for testing embedded software by applying a technique that largely automates the generation of test data while still enabling domain specialists to contribute with their knowledge and experience. The fitness function is calculated from a set of quality objective scores for a candidate solution, and a set of weights for those objectives is provided by the user which are combined into a single fitness. Thus, to guide the search, the domain specialist decides the relative importance of the quality objectives. In the experiments, the Differential Evolution was employed as search engine. An industrial evaluation was conducted, and results showed that the tool complements existing testing meth-

ods, given that the user interaction is essential in developing interesting and useful test cases.

As presented in Sect. 3.2, this work considers to be of great value three major questions when defining an Interactive Modeling for a search problem. Thus, Table 8 presents these three aspects under the perspective of the related work discussed above.

5.3 Treating human fatigue

In Kamalian et al. (2006) the use of artificial intelligence techniques is proposed to predict human evaluations based on previous interactions, analogously to the learning model proposed in this work. It is proposed the use of fuzzy inference systems and machine learning techniques to reduce human fatigue. The empirical evaluation was conducted under the context of microelectromechanical systems, also known as micromachines design problem. The fuzzy-system-based predictor was based on four system rules manually derived from the observation of previous human user tests, reaching a reduction in human effort of 51 % in average. Regarding the machine learning, it was investigated the results of four different approaches, and it was achieved, in average, a reduction in human effort of 31 %. These approaches achieved good accuracy on validation tests, but because of the great diversity in user scoring behavior, they were unable to achieve equivalent results on the user test data.

In Hsu and Huang (2005), it is argued that one of the main causes of human fatigue is the fact that there are occasions where the result preferred by the user does not exist in the search space. If it is not possible to ensure that the solution idealized by the user exists in the search space, the search process becomes more difficult and, consequently, the fatigue increases. Given this context, an effective method to create a search space that meets the customer values (or objectives) is proposed. It integrates the search space into a customer values-based IEC model to reduce user burden when designing their preferred products. A case study involving the design of water bottles was analyzed in order to verify the model's performance. Overall, the results confirm that a correct search space contributes to reduce the user fatigue.

Another alternative to ameliorate human fatigue that can be mentioned is the work in Wang et al. (2006). Fundamentally, it is proposed the creation of an absolute scale to improve the prediction of human evaluations in IEC, accelerating the algorithm convergence, and reducing the amount of human intervention. Thus, a concrete predictor method of mapping relative data to the absolute scale is proposed. Regarding the experiments, three methods were compared: an IGA with the proposed predictor using an absolute scale, an IGA with a conventional predictor using a relative scale, and an IGA without a predictor. First, the effectiveness of this method was evaluated using seven benchmark functions instead of a human user, which was aimed at verifying the convergence speed of an IEC using the proposed absolute rating. Next, the method was assessed through a subjective test using an IEC based individual emotion retrieval system in order to prove that the proposed predictor is effective in reducing the user fatigue. The proposed predictor using absolute evaluation had better prediction performance than conventional predictors, resulting in faster convergence.

Table 8 Interactive modeling to the related work

Paper	Problem	At which moment are the preferences from the DM captured?	What type and which preferences are provided to the search process?	How the preferences are incorporated and influence the search process?
Araújo et al. (2014)	Next release problem	Interactive	Implicit. Evaluation about the solution requirements selection	Another objective to be maximized
Dantas et al. (2015)	Release planning	A priori and Interactive	Explicit. Specific requirements allocation in releases	Another objective to be maximized
Tonella et al. (2013)	Requirements prioritization	Interactive	Explicit. Pairwise comparison between requirements	Minimize the number of pairwise comparisons
Bavota et al. (2012)	Software re-modularization	Interactive	Explicit. Add penalties for artifacts that are not where they should be	Soft constraint in the fitness function
Ghannem et al. (2013)	Refactoring	Interactive	Implicit. Rating the proposed refactorings	User rating combined with structural measurements
Simons et al. (2014)	Software design	Interactive	Implicit Rating candidate designs based on UML classes	Influence in the heuristic information of a iACO
Marculescu et al. (2015)	Test data generation	Interactive	Explicit. User decides the relative importance of the quality objectives	Weighting the different objectives to be optimized

6 Conclusions

Selecting requirements for the next release is a complex task in the incremental and iterative software development model, given the high number of combinations, technical constraints, multiple objectives and different stakeholders. An interesting approach to deal with this problem is the interactive optimization, which is a research field that uses the human tacit knowledge in computational search. The use of such optimization strategy is primarily recommended when the human influence can effectively contribute to the search process enabling the support decision system to absorb the user's implicit knowledge without the requirement to formalise this information a priori.

An initial proposal of the presented architecture has already been introduced in Araújo et al. (2014), in which preliminary results shown that an IGA can successfully incorporate the user preferences in the final solution. The present work significantly extends the previous work, improving both the architecture and empirical study. Through the combination of the benefits achieved by the Interactive Optimization, Machine Learning and SBSE, the main objective of this paper is to thoroughly present and evaluate the architecture to solve an interactive version of the NRP that allows an inclusion of human knowledge during the IGA evolution. Aiming to not overload the Decision Maker with a large number of interactions, a learning model was proposed to learn the human behavior and, eventually, replace the DM in the remainder of the evolutionary process. Thus, the architecture is composed by three different components with distinct responsibilities: (a) interactive genetic algorithm, (b) Interactive Module and (c) learning model. These components communicate among themselves and each one was properly modelled under the Interactive Next Release Problem (iNRP) perspective. Regardless of how many stakeholders are involved in the project, the approach focuses in interacting with only one crucial DM that needs to be fully immersed in the high-level project particularities and responsible to make the last call.

This paper considers both explicit (intrinsic characteristics of the problem) and implicit preferences (tacit and difficult to previously articulate) in the evaluation function. The first one is captured through the $score$ function that guides the search in order to achieve solutions that maximize the overall stakeholders satisfaction, in other words, selecting the requirements they want to be implemented in the next release. On the other hand, the implicit preference is gathered when the DM provides SE to each solution during the algorithm evolution. However, it is reasonable to have some trade-off between these objectives, because the importance assigned by a specific stakeholder to a certain requirement may not agree with the DM's broad view of the project. According to the specific scenario and project needs, it is possible to balance this trade-off defining the influence of each objective in the search process through the weights α and β specified in the architectural settings.

Two experiments were performed for the empirical study conducted in this work: the Artificial Experiment was verified with several and exhaustive scenarios, and the Participant-based Experiment investigated the feasibility of the architecture when it is used by software engineering practitioners. Through the analysis and results achieved in the empirical study, three main findings can be highlighted:

– The proposed architecture passes the sanity check when it demonstrates that as the number of interactions increase, more suitable to the DM preferences the final solution is;
– To incorporate the DM preferences, it is natural to have some *score* loss given the trade-off previously established. However, this *score* loss is considerably small, making the approach able to generate solutions as good as those produced in a fully automatic method, but satisfying most of the DM preferences;
– Even considering software engineering practitioners with different evaluation profiles, the proposed approach can improve their subjective satisfaction in comparison with a non-interactive solution;

Therefore, some interesting benefits of interactively including the DM in the next release planning during the evolutionary process can be underlined: (i) the DM could receive some feedback from the search through the presentation of the most promising solutions throughout the algorithm evolution; (ii) the capability to incorporate the changes from the DM's criteria during the search process; (iii) the DM may get some insights about the problem, and even dynamically adapt the decision criteria; and (iv) mitigate the feeling of intellectual exclusion by the user in the analysis, which may cause resistance and lack of confidence in the final result.

As future work directions, it is expected to provide other analyses regarding the learning model performance; develop a strategy to measure the increase or decrease of fatigue; interact with multiple decision makers where their subjective evaluations are considered as different objectives to be optimized in a many-objective paradigm; assess different ways for the DM to evaluate the solutions and suit the architecture for a interactive multi-objective next release problem (iMONRP) formulation aiming to maximize both *score* and SE, while minimizing the cost; and finally, conduct a extensive empirical study considering other traditional problems exploited in SBSE.

Acknowledgments The authors would like to thank the editorial staff and the anonymous reviewers for their professional and constructive comments, the participants of the experiment for their availability and, finally, the members of the Optimization in Software Engineering Group of the State University of Ceará.

References

Aljawawdeh, H.J., Simons, C.L., Odeh, M.: Metaheuristic design pattern: Preference. In: Proceedings of the Companion Publication of the 2015 on Genetic and Evolutionary Computation Conference, pp. 1257–1260. ACM (2015)

Araújo, A.A., Paixão, M.H.E.: Machine learning for user modeling in an interactive genetic algorithm for the next release problem. In: Proceedings of the 6th International Symposium on Search-Based Software Engineering (SSBSE '14), vol. 8636, pp. 228–233. Springer, Fortaleza, Brazil (2014). doi:10.1007/978-3-319-09940-8_16

Arcuri, A., Briand, L.: A hitchhiker's guide to statistical tests for assessing randomized algorithms in software engineering. Softw. Test. Verif. Reliab. **24**(3), 219–250 (2014)

Arcuri, A., Briand, L.C.: A practical guide for using statistical tests to assess randomized algorithms in software engineering. In: Proceedings of the 33rd International Conference on Software Engineering (ICSE '11), pp. 1–10. IEEE, Honolulu, HI, USA (2011). doi:10.1145/1985793.1985795

Arcuri, A., Fraser, G.: On parameter tuning in search based software engineering. In: Proceedings of the 3rd International Symposium on Search Based Software Engineering (SSBSE '11), vol. 6956, pp. 33–47. Springer, Szeged, Hungary (2011). doi:10.1007/978-3-642-23716-4_6

Bagnall, A.J., Rayward-Smith, V.J., Whittley, I.M.: The next release problem. Inf. Softw. Technol. **43**(14), 883–890 (2001). doi:10.1016/S0950-5849(01)00194-X

Baker, P., Harman, M., Steinhöfel, K., Skaliotis, A.: Search based approaches to component selection and prioritization for the next release problem. In: Proceedings of the 22nd IEEE International Conference on Software Maintenance (ICSM '06), pp. 176–185. IEEE, Philadelphia, Pennsylvania (2006). doi:10.1109/ICSM.2006.56

Bavota, G., Carnevale, F., Lucia, A.D., Penta, M.D., Oliveto, R.: Putting the developer in-the-loop: An interactive ga for software re-modularization. In: Proceedings of the 4th International Symposium on Search Based Software Engineering (SSBSE '12), vol. 7515, pp. 75–89. Springer, Riva del Garda, Italy (2012). doi:10.1007/978-3-642-33119-0_7

Carlshamre, P., Sandahl, K., Lindvall, M., Regnell, B., et al.: An industrial survey of requirements interdependencies in software product release planning. In: Proceedings Fifth IEEE International Symposium on Requirements Engineering, 2001, pp. 84–91. IEEE (2001)

Cho, S.B.: Towards creative evolutionary systems with interactive genetic algorithm. Appl. Intell. **16**(2), 129–138 (2002)

Dantas, A., Yeltsin, I., Araújo, A.A., Souza, J.: Interactive software release planning with preferences base. In: Proceedings of the 7th International Symposium on Search-Based Software Engineering (SSBSE '15), pp. 341–346. Springer, Bergamo, Italy (2015). doi:10.1007/978-3-319-22183-0_32

de Barros, M.O, Neto, A.C.D.: A survey of empirical investigations on ssbse papers. In: Proceedings of the 3rd International Symposium on Search Based Software Engineering (SSBSE '11), vol. 6956, pp. 268–268. Springer, Szeged, Hungary (2011). doi:10.1007/978-3-642-23716-4_24

do Nascimento Ferreira, T., de Souza, J.T.: An aco approach for the next release problem with dependency among requirements. In: Proceedings of the 3rd Brazilian Workshop on Search-Based Software Engineering (WESB '12). Natal, RN, Brazil (2012)

del Sagrado, J., del Águila, I.M., Orellana, F.J.: Ant colony optimization for the next release problem—a comparative study. In: Proceedings of the 2nd International Symposium on Search Based Software Engineering (SSBSE '10), pp. 67–76. IEEE, Benevento, Italy (2010). doi:10.1109/SSBSE.2010.18

Ferrucci, F., Harman, M., Sarro, F.: Search-based software project management. In: Software Project Management in a Changing World, pp. 373–399. Springer (2014). doi:10.1007/978-3-642-55035-5_15

Ghannem, A., Boussaidi, G.E., Kessentini, M.: Model refactoring using interactive genetic algorithm. In: Proceedings of the 5th International Symposium on Search Based Software Engineering (SSBSE '13), vol. 8084, pp. 96–110. Springer, St. Petersburg, Russia (2013). doi:10.1007/978-3-642-39742-4_9

Glass, R.L.: Facts and Fallacies of Software Engineering. Addison-Wesley Professional, Boston (2002)

Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., Witten, I.H.: The weka data mining software: an update. SIGKDD Explor. Newsl. **11**(1), 10–18 (2009). doi:10.1145/1656274.1656278

Han, J., Kamber, M., Pei, J.: Data Mining: Concepts and Techniques. Elsevier, Amsterdam (2011)

Harman, M.: The current state and future of search based software engineering. In: Proceedings of International Conference on Software Engineering / Future of Software Engineering 2007 (ICSE/FOSE '07), pp. 342–357. IEEE, Minneapolis, Minnesota, USA (2007). doi:10.1109/FOSE.2007.29

Harman, M.: Search based software engineering for program comprehension. In: Proceedings of the 15th IEEE International Conference on Program Comprehension (ICPC '07), pp. 3–13. IEEE, Banff, Alberta, Canada (2007). doi:10.1109/ICPC.2007.35

Harman, M., Clark, J.A.: Metrics are fitness functions too. In: Proceedings of the 10th IEEE International Symposium on Software Metrics (METRICS '04), pp. 58–69. IEEE, Chicago, USA (2004). doi:10.1109/METRIC.2004.1357891

Harman, M., McMinn, P., de Souza, J.T., Yoo, S.: Search based software engineering: techniques, taxonomy, tutorial. Empir. Softw. Eng. Verif. **7007**, 1–59 (2012). doi:10.1007/978-3-642-25231-0_1

Haykin, S.S.: Redes Neurais. Bookman, Porto Alegre (2001)

Hsu, F.C., Huang, P.: Providing an appropriate search space to solve the fatigue problem in interactive evolutionary computation. New Gener. Comput. **23**(2), 115–127 (2005)

Jiang, H., Zhang, J., Xuan, J., Re, Z., Hu, Y.: A hybrid aco algorithm for the next release problem. In: Proceedings of the 2nd International Conference on Software Engineering and Data Mining (SEDM '10), pp. 166–171. IEEE, Chengdu, China (2010). http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5542931

Kamalian, R., Yeh, E., Zhang, Y., Agogino, A.M., Takagi, H.: Reducing human fatigue in interactive evolutionary computation through fuzzy systems and machine learning systems. In: 2006 IEEE International Conference on Fuzzy Systems, pp. 678–684. IEEE (2006)

Karim, M.R., Ruhe, G.: Bi-objective genetic search for release planning in support of themes. In: Proceedings of the 6th International Symposium on Search-Based Software Engineering (SSBSE '14), vol. 8636, pp. 123–137. Springer, Fortaleza, Brazil (2014). doi:10.1007/978-3-319-09940-8_9

Marculescu, B., Feldt, R., Torkar, R.: A concept for an interactive search-based software testing system. In: Proceedings of the 4th International Symposium on Search Based Software Engineering (SSBSE '12), vol. 7515, pp. 273–278. Springer, Riva del Garda, Italy (2012). doi:10.1007/978-3-642-33119-0_21

Marculescu, B., Feldt, R., Torkar, R.: Objective re-weighting to guide an interactive search based software testing system. In: Proceedings of the 12th International Conference on Machine Learning and Applications (ICMLA '13), pp. 102–107. IEEE, Miami, Florida, USA (2013). doi:10.1109/ICMLA.2013.113

Marculescu, B., Feldt, R., Torkar, R., Poulding, S.: An initial industrial evaluation of interactive search-based testing for embedded software. Appl. Soft Comput. **29**, 26–39 (2015). doi:10.1016/j.asoc.2014.12.025

Marculescu, B., Poulding, S., Feldt, R., Petersen, K., Torkar, R.: Tester interactivity makes a difference in search-based software testing: A controlled experiment. arXiv preprint arXiv:1512.04812 (2015)

Miettinen, K.: Nonlinear Multiobjective Optimization, vol. 12. Springer, Norwell (1999)

Miettinen, K., Hakanen, J., Podkopaev, D.: Interactive nonlinear multiobjective optimization methods. In: Multiple Criteria Decision Analysis, pp. 927–976. Springer (2016)

Mitchell, T.M.: Machine Learning, 1st edn. McGraw-Hill Inc, New York (1997)

Palit, A.K., Popovic, D.: Computational intelligence in time series forecasting. Theory Eng. Appl. (2006). doi:10.1007/1-84628-184-9

Parmee, I., Hall, A., Miles, J., Noyes, J., Simons, C., et al.: Discovery in design: Developing a people-centred computational approach. In: DS 36: Proceedings DESIGN 2006, the 9th International Design Conference, Dubrovnik, Croatia (2006)

Piegat, A., Sałabun, W.: Nonlinearity of human multi-criteria in decision-making. J. Theor. Appl. Comput. Sci. **6**(3), 36–49 (2012)

Pitangueira, A.M.: Incorporating preferences from multiple stakeholders in software requirements selection an interactive search-based approach. In: 2015 IEEE 23rd International Requirements Engineering Conference (RE), pp. 382–387. IEEE (2015)

Pitangueira, A.M., Maciel, R.S.P., de Oliveira Barros, M.: Software requirements selection and prioritization using sbse approaches: a systematic review and mapping of the literature. J. Syst. Softw. **103**, 267–280 (2015). doi:10.1016/j.jss.2014.09.038

R-Project: http://www.r-project.org/ (2014). Accessed Apr 2016

Rousseeuw, P.J.: Least median of squares regression. J. Am. Stat. Assoc. **79**(388), 871–880 (1984)

Schachter, D.L.: Implicit memory: history and current status. J. Exp. Psychol. **13**(3), 501–518 (1987)

Semet, Y.: Interactive evolutionary computation: a survey of existing theory. University of Illinois, (2002)

Shackelford, M.: Implementation issues for an interactive evolutionary computation system. In: Proceedings of the 9th annual conference companion on Genetic and evolutionary computation, pp. 2933–2936. ACM (2007)

Shackelford, M., Corne, D.: A technique for evaluation of interactive evolutionary systems. In: Adaptive Computing in Design and Manufacture VI, pp. 197–208. Springer (2004)

Simons, C.: Interactive evolutionary computing in early lifecycle software engineering design. Ph.D. thesis, University of the West of England (2011)

Simons, C.L., Parmee, I.C.: Dynamic parameter control of interactive local search in uml software design. In: 2010 IEEE International Conference on Systems Man and Cybernetics (SMC), pp. 3397–3404. IEEE (2010)

Simons, C.L., Parmee, I.C.: Elegant object-oriented software design via interactive, evolutionary computation. IEEE Trans. Syst. Man Cybern. Part C **42**(6), 1797–1805 (2012). doi:10.1109/TSMCC.2012.2225103

Simons, C.L., Parmee, I.C., Gwynllyw, R.: Interactive, evolutionary search in upstream object-oriented class design. IEEE Trans. Softw. Eng. **36**(6), 798–816 (2010). doi:10.1109/TSE.2010.34

Simons, C.L., Smith, J.: A comparison of meta-heuristic search for interactive software design. Soft Comput. **17**(11), 2147–2162 (2013). doi:10.1007/s00500-013-1039-1

Simons, C.L., Smith, J., White, P.: Interactive ant colony optimization (iaco) for early lifecycle software design. Swarm Intell. **8**(2), 139–157 (2014)

Simons, C.L., Smith, J., White, P.: Interactive ant colony optimization (iaco) for early lifecycle software design. Swarm Intell. **8**(2), 139–157 (2014). doi:10.1007/s11721-014-0094-2

Takagi, H.: Interactive evolutionary computation: System optimization based on human subjective evaluation. In: IEEE International Conference on Intelligent Engineering Systems (INES'98), pp. 17–19 (1998)

Takagi, H.: Interactive evolutionary computation: fusion of the capabilities of ec optimization and human evaluation. Proc. IEEE **89**(9), 1275–1296 (2001)

Tonella, P., Susi, A., Palma, F.: Using interactive ga for requirements prioritization. In: Proceedings of the 2nd International Symposium on Search Based Software Engineering (SSBSE '10), pp. 57–66. IEEE, Benevento, Italy (2010). doi:10.1109/SSBSE.2010.17

Tonella, P., Susi, A., Palma, F.: Interactive requirements prioritization using a genetic algorithm. Inf. Softw. Technol. **55**(1), 173–187 (2013). doi:10.1016/j.infsof.2012.07.003

van den Akker, J., Brinkkemper, S., Diepen, G., Versendaal, J.: Determination of the next release of a software product: an approach using integer linear programming. In: Proceeding of the 11th International Workshop on Requirements Engineering: Foundation for Software Quality (REFSQ '05). Porto, Portugal (2005). http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.85.27%95

Wang, S., Wang, X., Takagi, H.: User fatigue reduction by an absolute rating data-trained predictor in IEC. In: IEEE Congress on Evolutionary Computation, 2006. CEC 2006, pp. 2195–2200. IEEE (2006)

Witten, I.H., Frank, E.: Data Mining: Practical Machine Learning Tools and Techniques. Morgan Kaufmann, Amsterdam (2005)

Xuan, J., Jiang, H., Ren, Z., Luo, Z.: Solving the large scale next release problem with a backbone based multilevel algorithm. IEEE Trans. Softw. Eng. **38**(5), 1195–1212 (2012). doi:10.1109/TSE.2011.92

Zhang, D., Tsai, J.J.: Machine learning and software engineering. Softw. Qual. J. **11**(2), 87–119 (2003)

Zhang, Y., Finkelstein, A., Harman, M.: Search based requirements optimisation: existing work and challenges. In: Proceedings of the 14th International Working Conference, Requirements Engineering: Foundation for Software Quality (RefsQ '08), vol. 5025, pp. 88–94. Springer, Montpellier, France (2008). doi:10.1007/978-3-540-69062-7_8