A Systematic Review of Interaction in Search-Based Software Engineering

Aurora Ramírez, José Raúl Romero, *Member, IEEE*, and Christopher L. Simons

Abstract—Search-Based Software Engineering (SBSE) has been successfully applied to automate a wide range of software development activities. Nevertheless, in those software engineering problems where human evaluation and preference are crucial, such insights have proved difficult to characterize in search, and solutions might not look natural when that is the expectation. In an attempt to address this, an increasing number of researchers have reported the incorporation of the 'human-in-the-loop' during search and interactive SBSE has attracted significant attention recently. However, reported results are fragmented over different development phases, and a great variety of novel interactive approaches and algorithmic techniques have emerged. To better integrate these results, we have performed a systematic literature review of interactive SBSE. From a total of 669 papers, 26 primary studies were identified. To enable their analysis, we formulated a classification scheme focused on four crucial aspects of interactive search, i.e., the problem formulation, search technique, interactive approach, and the empirical framework. Our intention is that the classification scheme affords a methodological approach for interactive SBSE. Lastly, as well as providing a detailed cross analysis, we identify and discuss some open issues and potential future trends for the research community.

Index Terms—Search-based software engineering, interaction, systematic literature review, optimization

1 Introduction

The design and development of complex, large-scale software systems can be non-trivial and challenging for the software engineer to perform. In an attempt to assist him/her, formulating software development activities as optimization problems has enabled the application of a range of metaheuristic search approaches. Such search-based software engineering (SBSE) [1] approaches have attracted significant research attention in recent years. Indeed, attempts have been made to support the software engineer in many cognitively challenging development activities by applying SBSE approaches across a range of lifecycle activities, as surveyed by Harman et al. [2].

1.1 Problem Description

The application of search-based approaches in support of software engineers raises a number of challenges. For example, it is generally difficult to formulate both a solution representation and a fitness measure of appropriate fidelity to fully reflect the reality of the software engineer's development activity [3]. This can be especially challenging when development decisions involve a range of seemingly unrelated and disparate criteria for solution acceptance where many cannot be explicitly articulated [4], [5]. Also, it is important that the run-time performance of the search approach is satisfactory with respect to support for development. Given the typical dimensionality and scale of search-based problems in the software development lifecycle, this may also be challenging [6] (chapter 9).

In addition, results of automated search approaches should engender the trust and acceptance of software engineers. As has been pointed out previously by Klien et al. [7], the software engineer and any computationally intelligent tool must use common ground to work jointly to agreed goals. Klien et al. suggest that *"to be a team player, an intelligent agent, like a human, must be reasonably predictable and reasonably able to predict others actions"*. Without this common ground, it is possible that software engineers may be reticent to apply probabilistic search approaches they cannot control and that produce solutions that do not look *"human-written"* [8]. In an example of search-based automatic software repair using genetic programming [9], the authors report that *"an additional challenge is establishing the credibility of automated repairs in terms of programmers confidence in them and their understandability"*. It is possible to speculate that because of these challenges, and because SBSE is a relatively recent field of research, SBSE does not yet appear to have provoked broad industrial adoption by software engineers.

To address the challenges of applying search to the realities of software development, as well as search performance and trust, attempts have been made to engage the software engineer by incorporating their participation during search. Exploiting software engineer insight via interaction within search can enrich fitness measure fidelity, and assist search performance by steering the trajectory of search to preferred

- A. Ramírez and J. R. Romero are with the Department of Computer Science and Numerical Analysis, University of Córdoba, Córdoba 14071, Spain. E-mail: {aramirez, jrromero}@uco.es.
- C. L. Simons is with the Department of Computer Science and Creative Technologies, University of the West of England, Bristol BS16 1QY, United Kingdom. E-mail: chris.simons@uwe.ac.uk.

Manuscript received 2 Apr. 2017; revised 11 Jan. 2018; accepted 1 Feb. 2018. Date of publication 5 Feb. 2018; date of current version 21 Aug. 2019. (Corresponding author: José Raúl Romero.) Recommended for acceptance by G. Fraser. For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below. Digital Object Identifier no. 10.1109/TSE.2018.2803055

regions, as well as guiding parameters of the search process itself. The notion of exploiting human interaction in search is not new, however. Early examples of human interaction with computational evolution can be found in Dawkins' biomorphs [10] and Sims' evolutionary virtual creatures [11], while a survey of interactive evolutionary computation was conducted by Tagaki in 2001 [12]. Later, in 2007, Branke et al. report on research into interactive multi-objective optimization [13].

In essence, notice that any attempt to involve the human in the search process with the aim of adapting the results to his/her preferences can be viewed as a sort of interaction [3]. Different approaches involve the capture of human preference *a priori* to set parameters and constraints for subsequent search, while other approaches allow search to reach termination and, *a posteriori*, present candidate solutions to the human for inspection [14]. However, in defining the scope of the problem description, we exclude *a priori* and *a posteriori* approaches and focus solely on direct human participation in search, which allows the user to intervene more than once and therefore react to intermediate results.

Nevertheless, while artificial intelligence (AI) tools can support optimization or knowledge discovery activities, it is possible that they can also introduce new types of errors. According to Lyell and Coiera [15], *"automation bias (AB) happens when users become overreliant on decision support, which reduces vigilance in information seeking and processing"*. The notions of automation bias and user complacency in automated decision support have been reported previously [16], [17], and experiments conducted by Bahner et al. [18] indicate that the perception of false recommendations is associated with high levels of user complacency. Shackelford [19] reports that an over-reliance on repeated 'human-in-the-loop' evaluation and interaction in search can result in user fatigue, wherein a non-linearity of user focus results in inconsistent search trajectory. Lyell and Coiera [15] suggest that strategies to minimize automation bias might focus on cognitive load reduction.

It can also be challenging to capture subjective evaluation of qualitative factors as a fitness measure in metaheuristic search. Indeed, there are some ill-defined aspects of good software systems quality that cannot be articulated, but nevertheless 'you know it when you see it'. This phenomenon has been referred to as the *"quality without a name"* [20], and is endorsed by the software design patterns community [21]. A distinction between implicit versus explicit user feedback and evaluation is drawn by Aljawadeh et al. in the metaheuristic design pattern 'preference' [14].

1.2 Motivation

Reflecting on SBSE in an overview of approaches to realizing artificial intelligence in software engineering (SE), Harman [22] notes a number of research gaps and challenges for future research. Among these are a need for SBSE to provide insight for the software developer, and novel 'AI-friendly' software development incorporating developer interaction to foster engagement, trust, comfort and satisfaction. Addressing these research challenges, interactive approaches have emerged using a variety of metaheuristics including evolutionary computing [S1] and swarm intelligence [S8], and interactive evaluation mechanisms including, for example, weights [S9], scores [S24] and rankings [S25]. We speculate that this could reflect the emergence of a new subfield for SBSE, namely interactive SBSE (iSBSE). We are first motivated to conduct a systematic review to connect diverse studies that have hitherto largely been examined separately, thus being able to expose current approaches, open issues and future trends. Second, this review can provide some guidance to researchers on the development of interactive SBSE approaches, whose main components will be categorized here. We also seek to provide an understanding of the emerging subfield of iSBSE as a prerequisite and promoter of further industrial adoption of SBSE.

1.3 Approach and Contribution

To make connections among the diverse areas of knowledge in interactive SBSE and offer an up-to-date comprehensive picture of the state-of-the-art, this paper offers a systematic literature review (SLR) of interactive SBSE. Drawing upon established, evidence-based software engineering systematic literature methodology [23], we locate and examine 669 papers published in conference and journal papers related to interactive SBSE. We define appropriate inclusion and exclusion criteria to select a set of 26 primary sources. We then analyze the content of the primary sources, and with reference to existing surveys in the field [2], [3], formulate a classification scheme to provide a comprehensive overview of contemporary research for interactive SBSE. In addition to quantitative analysis, we review qualitative aspects to identify trends, gaps, open issues and, in our opinion, significant future trends. As a result of this systematic literature review, the comprehensive knowledge provided can enable researchers to make informed and effective decisions regarding interactive SBSE, as appropriate to their problem context.

This SLR of iSBSE is posed under the formulation of the following four research questions (RQ):

RQ1: In what ways has interactivity been adopted within search-based software engineering? To answer this RQ, the proposed classification scheme serves to reflect the main characteristics of current developments in iSBSE.

RQ2: Which findings about search techniques and interactive approaches along the complete development cycle can be extracted from the current state-of-the-art? This RQ aims at bringing the current state of the use of interactive approaches in SBSE to light. To this end, primary studies are analyzed regarding the addressed SE problem, the proposed interactive mechanism and the defined experimentation framework.

RQ3: To what extent do the detected gaps hamper human interaction in SBSE? After thoroughly analyzing findings from RQ2, some gaps in the application of interactive approaches in SBSE can be identified. Then we speculate possible causes, as well as provide suggestions for improvement.

RQ4: What are the emerging trends and how might they be addressed in the future? Given that the combination of search-based software engineering and interactive optimization is still an emerging topic, this paper outlines potential future lines of research.

1.4 Organization

The remainder of the paper is organized thus. In Section 2, some background to search-based software engineering, search techniques and human interaction is provided. Next, Section 3 provides an overview of the review methodology used in this paper, including the classification scheme formulated for analysis of primary sources. In Section 4, we present findings of quantitative analysis before outlining the wider findings of the review in Section 5. In Section 6, we reveal the results of cross category data analysis to illustrate possible gaps and limitations of interactive SBSE, while speculating on possible causes. Then, in Section 7, we identify open issues and future trends before considering threats to validity in Section 8. Lastly, we conclude in Section 9.

2 Background

This section introduces the main concepts and terminology related to search-based software engineering, as well as a brief description of the most commonly applied search techniques in the field. Then, interactive optimization methods are presented.

2.1 Search-Based Software Engineering

The term search-based software engineering was coined by Harman and Jones in 2001 to encompass the application of search and optimization techniques to automate SE activities or support engineers during their resolution [2]. Search algorithms seek for optimal or near optimal solutions to a given problem, meaning that the first step towards the adoption of a SBSE approach is to reformulate the SE task as a search problem, i.e., the representation of the real-world problem so that the algorithm can generate and transform candidate solutions. In addition, the definition of a quantitative function, known as the fitness function, is required to let the algorithm discriminate between promising and poor solutions. This notion of quality is usually defined only in terms of software metrics, even though software engineers could make use of other more subjective mechanisms to assess quality, too.

Nowadays, SBSE covers all the phases of the software development process. In fact, there are a number of surveys and reviews specifically focused on each stage, including requirements management [24], software design [25], [26], software testing [27] and maintenance topics [28], [29], among others. For these specific SE problems, the selection of requirements to be implemented in the next iteration according to the stakeholders' interests and budgetary availability (the so-called next release problem, NRP), the early class analysis guided by software metrics, the automatic generation of test cases based on code coverage, or finding the optimal sequence of refactoring operations are some illustrative examples of SE tasks already solved using search techniques. The flourishing of these SLRs show an increasing interest in systematically analyzing the state of the field from diverse perspectives, though none of them address interactive optimization in SBSE.

The more complex the SE problem to be faced, the more sophisticated the optimization approach is required. Therefore, initial approaches adopted in early stages of SBSE were progressively enhanced to deal with multiple constraints, incorporate preferences, cope with uncertainty or allow the simultaneous optimization of two or more objectives [30]. Interestingly, Harman et al. [2] also suggest some areas of SBSE they consider overlooked and/or emerging. Among others, the potential of 'interactive optimization' is recognized, where issues related to possible fatigue and learning-effect bias still need to be thoroughly studied.

Continuing challenges make SBSE an increasingly active research field [31], which is also gaining the attention of industry [32]. Additionally, incorporating the software engineer's expertise to the process is essential to reach a good solution, which goes beyond the scope of software metrics [33]. Even for those cases where software metrics seem to be commonly accepted, the search process does not guarantee that the result of the search problem will result in representative solutions to the human, who may consider other additional, qualitative measures [34].

2.2 Search Techniques

Within the artificial intelligence field, search techniques are usually classified according to their ability to find the optimal solution and the exploitation of additional information when exploring the search space, i.e., the set of possible solutions [35]. Optimality of the final solution can be only guaranteed by exact methods, such as, for example, direct search of tree structures, dynamic programming, and integer linear programming. Therefore, to avoid having to explore all the candidate solutions, heuristic methods introduced the concept of problem-specific knowledge, defining rules to select or discard solutions depending on the current state of the process. Greedy search is a well-known example of a heuristic method.

Metaheuristic algorithms are iterative and non-exact methods, whose results could vary from one execution to another because of their stochastic nature. They are characterized by their efficiency and flexibility when solving complex optimization problems [36]. As opposed to heuristic procedures, metaheuristics are problem-independent techniques that define general and intelligent mechanisms to explore the search space, which are often inspired by biological processes. These algorithms are usually classified according to the number of candidate solutions they can handle at the same time [37]. Thus, a single-solution based metaheuristic starts from an initial solution and aims for its optimization in each iteration of the search process. On the other hand, a population-based metaheuristic deals with a set of solutions, combining their intrinsic information to reach a satisfactory solution. Within each category, a broad range of techniques can be found depending on how the search trajectory is determined or the nature-inspired mechanism that is being simulated by the algorithm.

Evolutionary algorithms (EAs) and Swarm Intelligence (SI) are two extensively used approaches of population-based metaheuristics. On the one hand, an EA starts with the random initialization of a population of solutions, called individuals. Then, an iterative process simulates the natural evolution of a species: selecting parents, generating offspring via crossover and mutation, and preserving the fittest individuals according to the fitness function. On the other hand, SI is based on emulating the collective behavior of entities working together in order to achieve a particular goal. For instance, a representative case is ant colony optimization (ACO), which takes inspiration from the foraging behavior of ants. Artificial ants look for the solution to the optimization problem as if they build the shortest path to the food source. Decisions on the most promising paths are based on a pheromone matrix that is dynamically updated considering areas already explored by other previous ants and the quality of their corresponding solutions.

A complementary categorization of search techniques relies on the nature of the optimization problem and, more specifically, on the number of objectives to be simultaneously handled. Thus, single-objective optimization problems are defined in terms of a unique objective, whereas multi-objective optimization problems are characterized by the presence of 2 or 3 objectives, often in conflict. Those problems requiring the definition of 4 or more objectives are currently recognized as many-objective [38]. This property affects the way a search algorithm solves the optimization problem. On the one hand, single-objective algorithms are able to directly compare candidate solutions using the fitness function. On the other hand, multi-objective algorithms [39] look for trade-offs between objectives and, consequently, require new strategies to decide which solutions should be kept in each step. Similarly, many-objective algorithms [38] still need to consider these trade-offs, though the problem complexity demands specific mechanisms to deal with the high dimensionality of the objective space.

2.3 Interactive Optimization

As described in [3], interactive optimization methods enable the user's active participation in the search process. Also known as 'human-in-the-loop' approaches, they are founded on three main pillars: the need for more realistic optimization models (with respect to problem representation and fitness function), the scope for improvement of the search efficiency, and the complete satisfaction of the user's expectations. Notice that the real problem under study often needs to be oversimplified in order to formulate its computational representation. This can limit the optimization method by making it unable to capture the complete decision context beforehand, e.g., when considering imprecise objectives or temporary constraints. Besides, users can often become frustrated and lack confidence in results because of the existing gap between the solutions automatically generated by optimization techniques and their realistic expectations. User participation is specially relevant in dynamic scenarios, where the expert's knowledge could be considered as a complementary source for the exploratory capability of the search procedure. However, other factors inherent in being human (e.g., uncertainty and fatigue) can appear and need to be thoroughly considered [40].

In order to involve the human in the search process, the optimization method is required to provide intermediate outcomes to enable him/her to better understand the current search state, as well as how the process itself is being conducted. The role of human consists in returning some feedback that will be considered somehow thereafter in the iterative process [3]. In this context, the taxonomy proposed by Meignan et al. [3] provides a primary source to properly classify current interactive approaches in SBSE. More precisely, Meignan et al. establish five categories of interactive methods:

- *trial and error,* consisting of interactive adjustment of parameters;
- *interactive reoptimization,* aiming at redefining the formulation of the optimization problem;
- *interactive multiobjective optimization,* wherein the user interactively determines an appropriate trade-off between objectives;
- *interactive evolutionary computation* (IEC) [12], in which the human serves as fitness function; and
- *human-guided search,* a local search procedure directly incorporating transformations approved by the human.

When designing these interactive methods, there are a number of additional requirements that need to be considered, such as the point in which interaction happens (e.g., occasional or periodical participation), the specific task undertaken by the human (e.g., comparison, evaluation or correction of solutions) or the nature of the information gathered from such interaction, among others. Although the taxonomy proposed by Meignan et al. is comprehensive and covers many general interactive methods, not all the additional requirements are considered. Therefore, it is necessary to adapt and extend the taxonomy to address the specific requirements of interactive SBSE studies.

Finally, it is worth mentioning that SBSE can also be applied in combination with other types of approaches such as machine learning and interactive methods, which is known as interactive machine learning [41]. In short, machine learning (ML) techniques typically seek to predict classifications of new data by learning from past experience. With user involvement, a ML algorithm is able to integrate additional valuable information in order to build more precise and realistic models while gaining further knowledge.

3 Review Methodology

The review method has been built upon *best practice* in systematic literature review [23], [42]. Recent SLRs in the fields of empirical software engineering [43] and search-based software engineering [26], [44] have been taken as a reference, too. This section explains in detail the methodology used for conducting this review on iSBSE. To address the precisely raised research questions, a method is defined and followed in order to determine how the literature revision is performed. Such a method includes mentioning the literature search strategy, explicitly enumerating the inclusion and exclusion criteria for the papers found, and determining how the review data are collected from the selected works.

3.1 Literature Search Strategy

All the revised papers have been queried from a wide range of scientific literature sources, to prevent relevant studies remaining hidden:

- *Digital libraries:* ACM Library, IEEE Xplore, ScienceDirect, SpringerLink.
- *Citation databases:* ISI Web of Knowledge, Scopus.
- *Citation search engines:* DBLP, Google Scholar.
- *Specialized sources:* SBSE repository maintained by the CREST research group at UCL,[1] as a reference for the SBSE community.

[1] http://crestweb.cs.ucl.ac.uk/resources/sbse_repository

TABLE 1
The Two Search Strings Defined

Search string #1	(interactivity OR interactive OR human-in-the-loop OR user-interaction OR user-centered OR user-centred) AND (search-based OR "search based") AND "software engineering"
Search string #2	(interactivity OR interactive OR human-in-the-loop OR user-interaction OR user-centered OR user-centred) AND software AND (requirements OR architecture OR design OR development OR testing OR debugging OR verification OR maintenance OR evolution) AND (search OR search-based OR optimisation OR optimization)

To build the search strings, the three authors first defined a list of terms embracing the variety of both application domains and search techniques to be covered. Synonyms and keywords were derived from this list and logical operators (AND, OR) used to properly link search terms. Beforehand, a pilot search was conducted on all the data sources to verify the adequacy and effectiveness of the resulting search strings. Then, some minor changes were applied to the queries to avoid a number of unwanted outcomes being returned. The refined search strings are shown in Table 1. Notice that search string #1 looks for SBSE publications in the broadest sense, whereas the search string #2 looks for those publications that do not explicitly contain the term SBSE but still apply search and optimization techniques to a SE task. The ACM Computing Classification System[2] and the IEEE Taxonomy[3] were used to define the list of tasks. When required, a search string was adapted to the specific query language used by a particular engine or database. Searches were conducted at first by the first author, and then double-checked by the rest of authors. String adjustments were agreed by all authors.

After the execution of the search queries, a total of 653 references were returned. Following best practices in conducting SLRs [23], [42], all authors performed independent manual searches to complement automatic results. More specifically, web profiles of relevant authors and their networks were consulted, and cross-references were checked following a *snowballing* procedure. As a result, 16 new references were added. Table 2 shows the number of publications found per data source, including the manual search. Then, an initial manual examination of titles and abstracts enabled publications not related to SBSE to be discarded. In this way, the list of candidate papers decreased to 67 publications (see Table 2). Finally, the papers to be included in the review, known as primary studies, are rigorously arrived at by applying the corresponding inclusion and exclusion criteria (see Section 3.2). A total number of 26 primary studies is obtained. During this search period monitoring and follow-up meetings were held to detect errors, disagreements or deviations from planned procedure.

3.2 Inclusion and Exclusion Criteria

From the list of candidate papers aforementioned, only those publications related to iSBSE could be considered as primary studies, i.e., the authors should propose an interactive search-based approach to solve a software engineering task. Such a limitation could be broken down into the following inclusion criteria

1) The search model should explicitly incorporate a user, who is required to perform at least one interaction in order to solve a task during the search process, e.g., evaluation or selection of candidate solutions. Another valid approach is the user to participate at least once the search has finished but only when another iteration or algorithm is executed afterwards, so that the user's opinion could be integrated somehow in the overall process.
2) A paper should describe the method followed for the user's interaction. It should explain the role of the human in the search process, how his/her feedback is then integrated into the search process, or any other relevant aspect concerning the interactive action.
3) The search problem should be defined in terms of a decision space, evaluation objectives and the existing constraints, if any. The problem could be formulated as either single-objective or multi/many-objective, or both.
4) The search problem should be framed within one or several phases of the software development life cycle.
5) The search technique applied to address the software engineering problem is not restricted, but it should be a computational method.
6) The nature of the information provided by the user is not restricted, but it should cause an effect on the search.
7) Theoretical proposals are allowed.

In contrast, papers meeting any of the following exclusion criteria were not considered as a primary study:

1) The interactive approach does not address a software engineering optimization problem.
2) The user is only able to (re)configure the parameters of the search algorithm right before/after an independent run, or he/she is only able to make a decision at the end of the process, e.g., by selecting one single individual from the resulting solution set

TABLE 2
Number of References per Data Source

Data source	Search results	Candidate papers	Primary studies
ACM Library	86	11	4
DBLP	20	15	5
IEEE Xplore	122	15	5
ISI-WoK	207	21	10
SBSE Repository	33	20	10
ScienceDirect	43	5	4
Scopus	218	32	13
SpringerLink	124	15	7
Manual search	16	15	6
Total unique papers	669	67	26

[2] https://www.acm.org/publications/class-2012
[3] https://www.ieee.org/documents/taxonomy_v101.pdf

generated after completing a multi-objective optimization algorithm.

3) The research paper is written in a language other than English.
4) The full text of the manuscript is not accessible.
5) Either the publication process has not followed an accurate scientific peer-review process or there is no clear evidence of this point.

Finally, if multiple variants of the same research work are found, the conditions under which a paper is considered as a primary study are described next. They are applied if the same authors have published different papers for the same interactive approach, so only significant contributions are analyzed for the review. Nonetheless, notice that these constraints are also influencing the statistical study shown in Section 4:

- Considering a previous primary study, if the problem is the same but the technique is different and novel, then accept.
- Considering a previous primary study, if both the problem and the technique are the same but the authors have reported different findings from a new significantly different experimentation, then accept.
- If a journal paper is found as an extension of a previous conference paper and it satisfies the inclusion criteria, then reject the previous conference paper unless it provides with different but significantly distinct experimental outcomes.

3.3 Data Collection Process

After identification, primary studies were thoroughly analyzed to rigorously conduct the review process. With this aim, these papers are randomly distributed among the authors following some basic rules: (a) only the authors (3) of this review could participate in data collection process; (b) every primary study should be scrutinized by at least two reviewers; (c) a paper could not be reviewed by any of its authors; (d) there should be a balance among reviewers with respect to the number of papers, their category (i.e., conference, journal, etc.) and their SBSE topics that they study. Each reviewer should compile a data extraction form for each primary study. After completing the analysis, these forms are brought together to detect any disagreements between the reviewers. If so, the author who did not participate in the data extraction of the corresponding paper should also read the primary study and provide an opinion before a final and collective decision is reached.

All papers have been objectively reviewed under strict control and consistent conditions. No information has been inferred during the extraction process, and authors of the primary studies were not contacted. In the case of a potential conflict of interest when a primary study was authored by a reviewer, she/he did not participate in the data extraction and analysis. In addition, missing data are possible for some categories, and are reflected accordingly.

At this point, it is worth highlighting that data extraction discrepancies appeared for many of the primary studies, most of which were minor disagreements that could be resolved after a brief discussion between the reviewers involved in the data extraction process. In general, discrepancies appeared in all the categories. However, the disagreements appearing most frequently were focused on very specific aspects. For example, characterizing the role of the user, the interaction mechanism (type of solution shown and frequency of interaction), the influence of the user's opinion and the scope of the case studies were the most contentious discussion points. We observed that deciding the role of the user and the influence of his/her opinion were too often subject to the reader's interpretation. We also observed that the description of information such as the interaction mechanism was incomplete or imprecise in most primary studies. Furthermore, the lack of agreed benchmarks and standards within the SBSE field often made difficult to determine the scope of the cases studies. This emphasizes the importance of following a strict conceptual framework and agreed guidelines when defining the interactive proposal, given that misunderstanding can be easily reached otherwise.

3.4 Classification Scheme

In order to facilitate systematic data collection, as a response to RQ1, we developed a classification scheme drawing on best practice [23]. This scheme classifies iSBSE data relating to the following aspects: (i) *meta-information*, (ii) *problem formulation*, (iii) *search technique*, (iii) *interactive approach* and (v) *experimental framework*. For each of these aspects, we determined a number of data categories, and enumerated discrete values for each one. The resulting classification scheme does not only provide the guidelines for reviewers during the data extraction process, but also is intended to assist those researchers interested in iSBSE to accurately define their new approaches.

First, meta-information related to the primary sources includes: (i) author(s) name(s); (ii) title of publication; (iii) type of publication (journal, conference or workshop); (iv) name of publication and publisher; (v) year of publication; and (vi) volume, issue and pages. For the sake of brevity, a brief introduction to the rest of categories follows below. Notice that a summary of the classification scheme can be found in the supplementary material, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TSE.2018.2803055, while a description is available on a website[4] accompanying this paper.

3.4.1 Classification of the Problem Formulation

The problem formulation refers to information describing the SE problem under study and its computational representation. Drawing on classifications used in previous surveys by Harman et al. [2] and Li et al. [38], and established software engineering texts [45], data are classified into 4 categories:

- *Type of software engineering problem*, which refers to the phase of the software lifecycle where the problem is framed including values such as *requirements*, *analysis and design*, etc.
- *Development practice*, which corresponds to the SE methodology in which the problem is contextualized (*prescriptive model*, *specialized model* or *agile development*), if any.

[4] http://www.uco.es/grupos/kdis/sbse/isbse

- *Number of objectives* stated in the definition of the problem. Depending on this number, the problem is classified as *single-objective* (1 objective), *multi-objective* (from 2 to 3) or *many-objective* (more than 3).
- *Constraints*, which indicates whether the optimization problem comprises a formal definition of constraints (i.e., *constrained problem*). If there is an explicit mention to the lack of constraints, it will be defined as an *unconstrained problem*. Otherwise, N/A (not applicable/not available) is specified.

3.4.2 Classification of the Search Technique

Drawing on previous surveys of optimization heuristics [3], [37] and established texts relating machine learning to software engineering [46], search technique data are classified along 2 categories:

- Search *algorithm type*, which describes the resolution technique applied to the problem. The specific values correspond to techniques like *exact procedure*, *heuristic procedure*, *metaheuristic procedure* and *machine learning*, as introduced in Section 2.2.
- *Objective approach*, which specifies the number of objectives with which the algorithm deals, independently of the number of objectives defining the SE problem. The possible values are *single/multi/many-objective*, where single-objective could also include the use of an aggregation fitness function.

3.4.3 Classification of the Interactive Approach

The taxonomy proposed by Meignan et al. [3] provides a baseline classification for the interactive approach, which has been significantly extended to include additional information specifically adapted to the iSBSE field. We formulated 12 categories of data specific to iSBSE, as follows.

- The *interactive algorithm type* (adapted from [3]) differentiates among four interactive optimization methods: (i) *interactive reoptimization*, where the problem definition is refined; (ii) *preference-based interactivity*, where the goal is to incorporate users preferences during the search; (iii) *human-based evaluation*, wherein the user totally or partially replaces the fitness function; and (iv) *human-guided search*, where user actions directly impact candidate solutions. These methods can appear in combination.
- Two non-exclusive *purposes of user interaction* are defined to describe the focus of such interaction [3]. On the one hand, an interaction is *problem-oriented* when the user contributes to (re)define the optimization problem. On the other hand, a *search-oriented* interaction is focused on actions conducted during the search procedure in order to improve its efficiency.
- The *role of the user* in the process depends on the specific selected purpose of interaction [3]. Hence, if it is a problem-oriented interaction, the user could be an *adjuster* of constraints and/or objectives, or and *enricher* who adds or removes them. For a search-oriented interaction, there are three possible roles: (i) *assistant*, which refers to the selection or modification of solutions; (ii) *guide*, which means that the user controls the search process; or (iii) *tuner*, relating to adjusting search parameters.
- The *type of user task* describes the action(s) performed by the user over the solution(s). It includes *evaluation* of some aspects of the solution quality, the *selection* of promising solutions, the *comparison* between two or more solutions, or even the manual *modification* of the solution, including freezing it. More than one task could be performed if requested.
- The *evaluation mechanism* should be identified if any sort of evaluation is involved in the interaction. The possibilities consist of (i) providing a *fitness* value, (ii) assigning *weights*, (iii) offering a discrete *score*, (iv) giving *rankings*, and promoting or demoting with a *reward or penalization*.
- The *adjustment of interaction time* describes how often interaction occurs. If interruptions are *fixed* a priori, an interaction could occur *every iteration*, *every N iterations* or *between two runs* of an algorithm. Otherwise, if the interaction is *dynamic*, then it could be either based on the course of the search (*adaptive*) or requested by the user (*on demand*).
- The *number of solutions* shown to the user per interaction can be set to only *one*, a *pair*, *N* solutions or *all* of them, i.e., the whole population.
- The *level of detail* indicates whether the information shown to the user about a given solution for a specific interaction represents a *complete* or a *partial* solution.
- The *selection strategy* for solutions to show to the user can rely on the algorithm (*fixed*) or the user (*free*). In the first case, the specific criterion is determined by selecting the *best*, *random* or a *specific* solution. It is also possible to show *all* of them.
- *Feedback integration* refers to the mechanism that manages the users opinion [3]. It is defined as *model-free* when the preference information is directly incorporated by the algorithm. In contrast, *model-based* approaches include a learning process instead.
- The *preference information lifetime* describes the temporal scope of users feedback [3]. Information lifetime can be *step based* (used between two sequential interactions, (ii) *short-term* (valid for a single run or (iii) *long-term* (reused across different executions).
- *Information validity* indicates how the information is integrated and kept during the search. Three values are possible: (i) *permanent*; (ii) *flexible*, if the user can modify feedback; and (iii) *unrestricted*, if the user can revoke feedback.

3.4.4 Classification of Experimental Frameworks

Categories regarding the experimental framework are extracted after a comprehensive examination of primary studies and for the statistical analysis, the guide by Arcuri and Briand [47] was considered. We formulated 8 categories of data to classify the experimental frameworks used in the primary studies, as follows.

- *Type of study*, which determines the scope of the experimental study presented in the paper. Four values have been identified: (i) *theoretical proposal*, (ii) *sample execution*, (iii) *simulated interaction* and (iv) *empirical investigation*.

TABLE 3
List of Variants of Primary Studies Sorted by Publication Year and First Author's Surname

Reference	Authors	Title	Journal/Conference	Year
[48]	A.A. Araújo and M.H. Paixão	Machine Learning for User Modeling in an Interactive Genetic Algorithm for the Next Release Problem	Symposium on Search Based Software Engineering	2014
[49]	B. Marculescu et al.	Practitioner-Oriented Visualisation in an Interactive Search-Based Software Test Creation Tool	Asia-Pacific Software Engineering Conference	2013
[50]	B. Marculescu et al.	A Concept for an Interactive Search-Based Software Testing System	Symposium Search Based Software Engineering	2012
[51]	P. Tonella et al.	Using Interactive GA for Requirements Prioritization	Symposium on Search-Based Software Engineering	2010
[52]	L. Troiano et al.	Interactive Genetic Algorithm for choosing suitable colors in User Interface	Learning and Intelligent Optimization Conference	2009
[53]	C. Simons and I.C. Parmee	Agent-based Support for Interactive Search in Conceptual Software Engineering Design	Genetic and Evolutionary Computation Conference	2008
[54]	C. Simons and I.C. Parmee	User-centered, Evolutionary Search in Conceptual Software Design	IEEE Congress on Evolutionary Computation	2008
[55]	N. Monmarché et al.	On Generating HTML Style Sheets with and Interactive Genetic Algorithm Based on Gene Frequencies	European Conference on Artificial Evolution	1999

- For empirical studies, the *number of participants* should be specified using prefixed intervals.
- For empirical studies, the *participant position* should be determined to develop a complete profile of the users. Accepted values include: (i) *engineer*, (ii) *PhD academia staff*, (iii) *postgraduate student*, (iv) *undergraduate student*, and (v) any *other*.
- For empirical studies, total work experience (*expertise*) of the users should be also gathered in terms of time intervals varying between *less than 5 years* and *more than 20 years*.
- Experimental *evaluation criteria* should be selected to indicate how results are analyzed. The three criteria found in iSBSE studies are *measures* (e.g., fitness values), examples of *solutions* and responses to *questionnaires*.
- *Case studies* reported in experiments can be artificially created (*synthetic*), small but real problems known to participants (*controlled environment*) or an *industrial case*.
- If *evidence* of additional materials is found, the available information should be accessed, compiled and classified by their respective types (*source code*, *problem instances*, *raw results*, *solutions*, *questionnaires*, *transcripts*, *additional statistics* or *other*).
- Studies including *statistical tests* may contain one or more types of tests (*pairwise comparison*, *multiple comparison*, *effect size measurement*).

4 Quantitative Analysis

As an initial result of the analysis conducted to respond RQ2, this section provides quantitative information about iSBSE studies and their respective authors in order to provide an overview of the state of the field.

4.1 Information on Sources

After strictly applying the method described in Section 3.1, 26 primary studies have been obtained (a separate list of references is provided at the end of the paper). It should be noted that the 8 candidate papers shown in Table 3 also satisfied the inclusion criteria, but they were considered as variants of primary studies according to the review protocol (see Section 3.2). Although excluded from the review analysis, they have been included within this quantitative analysis to reduce the bias towards certain types of publication and authors.

Fig. 1 depicts the bar chart displaying the cumulative number of publications per year. As can be observed, the first two contributions within the field were published in 1999 by Monmarché et al., though no more publications appeared until the mid 2000s. In the earlier years, the number of papers showed a small but steady increase. However, after 2013, iSBSE attracted increasing attention. This is evidenced by how the different types of publication have evolved. Fig. 1 reveals that the majority of publications (71 percent) are contributions to international conferences. Nevertheless, 29 percent of the total of publications refers to journal papers. It is worth noticing that 8 out the 10 journal papers were published after 2013.

Regarding the journals where papers are published and, consequently, their audience, submissions are equally

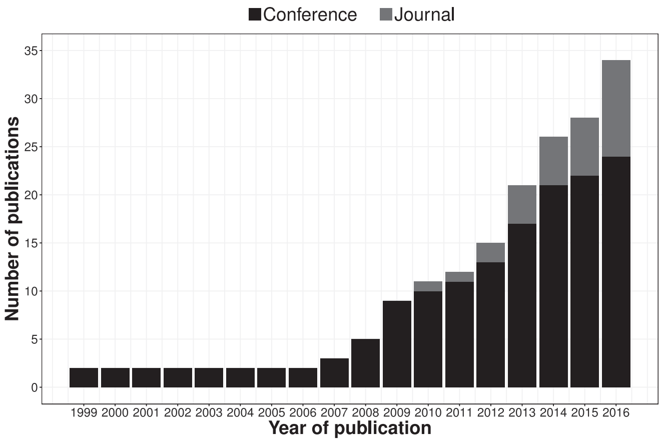


Fig. 1. Cumulative number of publications per year and type.

TABLE 4
List of Authors with Three or More Publications

Author	No. Publications
Allysson Araújo	4
Cosimo Birtolo	3
Altino Dantas	3
Robert Feldt	6
Marouane Kessentini	3
Bogdan Marculescu	5
Francis Palma	3
Ian Parmee	5
Christopher Simons	6
Jerffeson de Souza	3
Angelo Susi	3
Paolo Tonella	3
Richard Torkar	5
Luigi Troiano	3

TABLE 5
List of Countries

Country	No. Publ.	Publication year
Brazil	4	2014-2016
Canada	1	2013
China	2	2013, 2016
Egypt	1	2014
Finland	1	2013
France	2	1999
Ireland	1	2014
Italy	7	2009-2013, 2016
Singapore	1	2013
Sweden	6	2009, 2012, 2013, 2015, 2016
Tunisia	1	2014
United Kingdom	7	2008-2010, 2012, 2014, 2016
United States of America	7	2007, 2013, 2014, 2016

distributed, only *Applied Soft Computing* and *Information and Software Technology* appearing twice (see List of primary studies before References). Most of these articles are ranked top according to the *Journal Citation Reports*, covering a broad spectrum of areas: Software Engineering (3), Artificial Intelligence (2), Multidisciplinary Sciences (2) and Cybernetics (1). This seems to be clearly motivated by the multidisciplinary nature of SBSE. As for conference papers, a similar distribution between the SE and AI communities is found. However, it is worth mentioning a more frequent appearance of the *Symposium on Search-Based Software Engineering*, a specialized event where 3 papers were presented and extended afterwards as journal papers [S1], [S4], [S18].

4.2 Information on Authors

A total of 66 authors based in 13 different countries have been identified from the list of publications under study. Table 4 shows the most frequent authors in the area, including their number of co-authored publications. They contribute to 76 percent of primary studies and their variants. With respect to their country, Table 5 shows the total number of papers published in iSBSE per country. Authors based in Italy, Sweden, the United Kingdom (UK) and the United States of America (USA) are the most frequently published in iSBSE. With the aim of providing a baseline for SE publications, the affiliation countries and level of cooperation have been also analyzed from a renowned broad community like ICSE (International Conference on Software Engineering). Between 2007 and 2016, 845 ICSE technical papers were authored by researchers based in 41 different countries, USA, Canada and Germany being the most frequent. Both the number of papers and countries show that iSBSE is still a relatively small area, but somehow more widely distributed by country. For instance, around 50 percent of ICSE papers have one or more authors affiliated to a USA institution, whereas authors based in USA appear in 21 percent of iSBSE papers. With a similar percentage in iSBSE, UK and Italy obtain the 5th and 6th positions for ICSE publications, respectively. The level of cooperation between different countries is slightly lower than that of the ICSE community, notably because iSBSE is still in its earlier stages. In this sense, 53 percent of the considered publications are co-authored by researchers working for different institutions (51 percent in ICSE), 39 percent of which refer to collaborations between authors located in different countries (55 percent in ICSE).

5 Findings of the Review Process

Responding to RQ2, this section provides a detailed review of the primary works under study. Following the high-level structure specified in the classification scheme described in Section 3.4, the data extraction process has produced insightful results regarding the problem formulations addressed in iSBSE, the search techniques applied, the mechanisms for human interaction, and how outcomes are empirically validated.

5.1 Problem Formulation

As shown in Table 6, iSBSE has been proposed to address a significant number of project phases. Tasks related to the analysis and design, where creativity and experience play a key role, have emerged as problem fields where human interactivity has been applied to achieve better search performance. In these cases, there is a variety of design artifacts to be optimized, including object-oriented specifications [S8], [S10], [S11], [S13], software architectures [S9], software product lines [S7] and graphic user interfaces [S6], [S12], [S14]. In contrast, other problem areas like testing and verification or coding, incorporate the software engineer's opinion in the search process to a limited extent. Notice that interactive search-based refactoring is classified under the area 'distribution and maintenance', wherein artifacts include both design models and source code. Others, such as those related to project management, are as yet unexplored. Indeed, the lack of research studies into interactive search-based project management is perhaps surprising, given that others have previously explored interactive search-based approaches for project management more generally.

Only 15 percent of sources specify the *development practice* for interactive search (see Table 6). In such a case, an iterative and incremental process model is indicated for the majority of sources, all of them dealing with the next release problem [S1], [S2], [S3], while one source reports a specialized process for model driven engineering [S24]. Agile development is not specifically reported as a software engineering process in any source.

TABLE 6
Problem Formulation

Category	Percentage	Papers
Type of software engineering problem		
Requirements	19%	[S1], [S2], [S3], [S4], [S5]
Analysis and design	38%	[S6], [S7], [S8], [S9], [S10], [S11], [S12], [S13], [S14], [S15]
Code implementation	4%	[S16]
Testing and verification	12%	[S17], [S18], [S19]
Distribution and maintenance	27%	[S20], [S21], [S22], [S23], [S24], [S25], [S26]
Project management	0%	
Development practice		
Prescriptive process model	12%	[S1], [S2], [S3]
Specialized process model	4%	[S24]
Agile development	0%	
Not specified	85%	[S4], [S5], [S6], [S7], [S8], [S9], [S10], [S11], [S12], [S13], [S14], [S15], [S16], [S17], [S18], [S19], [S20], [S21], [S22], [S23], [S25], [S26]
Number of objectives		
Single-objective	27%	[S1], [S2], [S3], [S4], [S5], [S15], [S26]
Multi-objective	38%	[S6], [S8], [S10], [S11], [S12], [S13], [S14], [S19], [S20], [S23]
Many-objective	23%	[S7], [S9], [S17], [S18], [S24], [S26]
Not specified	15%	[S16], [S21], [S22], [S25]
Constraints		
Constrained	38%	[S1], [S2], [S3], [S4], [S5], [S8], [S10], [S11], [S13], [S22]
Unconstrained	27%	[S7], [S12], [S15], [S19], [S23], [S24], [S26]
Not specified	35%	[S6], [S9], [S14], [S16], [S17], [S18], [S20], [S21], [S25]

Regarding the *number of objectives* reported for each problem context, the majority (i.e., 62 percent) of sources describe a multi/many objective software engineering problem, compared to (27 percent) that report problems formulated as single objective. This finding is in line with the general trend of SBSE, where the adoption of multi-objective problem formulations is increasing in order to better reflect the multiple decision factors involved in any SE problem [30]. Besides, some studies i.e., [S21], [S22], [S25], [S16] are not sufficiently informative about the number of objectives, which is noteworthy in a field like SBSE. With respect to the applicability of *constraints* in the problem formulation, fewer than half of the sources (38 percent) can be properly described as constrained. Notice that the constrained problems mainly refer to the NRP [S1], [S2], [S3], [S4], [S5], in which budgetary restrictions need to be considered, and design tasks [S8], [S10], [S11], [S13], which usually define specific rules to derive valid specifications. The remainder may either be described as unconstrained, or the application of constraints is not mentioned in the source.

5.2 Search Technique

Table 7 classifies the primary studies according to the search methods they apply and objective approach (see Section 3.4). In terms of the *algorithm type* used as a search technique, it can be observed that metaheuristics are the most frequently applied (88 percent). Within this group, evolutionary computation is used by a large majority of primary sources (81 percent), though different types of evolutionary algorithms are considered, including genetic algorithms (57 percent), multi-objective evolutionary algorithms (24 percent), differential evolution (14 percent) and genetic programming (5 percent). This may largely be due to a historical bias to evolutionary computing among metaheuristics carrying through to SBSE [2]. Nevertheless, two studies based on the application of swarm intelligence follow the ACO paradigm. In the minority is one paper that uses exact search, and two that use machine learning. None of the primary studies rely on heuristics to conduct the search.

With respect to the *objective approach* of the search techniques described, the majority of sources (73 percent) describe

TABLE 7
Search Techniques Currently Used in iSBSE

	Single-objective	Multi-objective	Many-objective	Not applicable
Exact	[S5]			
Metaheuristic				
Single-solution based	[S20]			
Evolutionary computation	[S1], [S3], [S6], [S17], [S18], [S19], [S20], [S21], [S22], [S24] [S4], [S9], [S11], [S12], [S14], [S26]	[S10], [S23], [S26] [S11], [S13]	[S7]	[S15]
Swarm intelligence	[S2], [S8]			
Machine learning				
Supervised				[S16]
Unsupervised				[S25]

TABLE 8
Interactive Approaches and Their Application in iSBSE

Interactive algorithm type	Description	Percentage	Papers
Interactive reoptimization	Feedback is used to modify some elements of the problem formulation (adding constraints or objectives, adjusting weights, etc.) instead of the solution itself.	42%	[S2], [S3], [S4], [S5], [S7], [S9], [S10], [S17], [S18], [S19], [S20]
Preference-based interactivity	Preference information is used to guide the search towards some zones of search/objective space, e.g., selecting between two candidate solutions.	42%	[S7], [S11], [S13], [S14], [S15], [S16], [S20], [S21], [S23], [S25], [S26]
Human-based evaluation	Human subjective judgment is obtained by means of different evaluation mechanisms (e.g., scores, rankings, etc.)	27%	[S1], [S4], [S6], [S8], [S12], [S22], [S24]
Human-guided search	Actions performed by the human that have a direct impact on the solutions, such as freezing parts of the encoding.	8%	[S9], [S15]

a single-objective search approach, many of them using an aggregation of several metrics to compute fitness. The remainder describe multi/many-objective search, with the exception of one paper [S15] not defining a fitness function and ML approaches. As can be seen from Table 7, two primary studies [S26], [S11] explore the corresponding SE problem from two different perspectives. Even though most of the metaheuristic paradigms have been adapted to deal with multiple objectives, only evolutionary techniques have been applied in iSBSE hitherto, due presumably to their broad popularity and maturity.

5.3 Interactive Approach

First, we examined the *interactive algorithm type* used in the interactive approach with the software engineer. As shown in Table 8, at 42 percent, interactive reoptimization and preference-based interactivity are the most frequently occurring type of algorithm used. Human-guided evaluation is described in 27 percent, while human-guided search is employed in two papers. Notice that five primary studies [S20], [S7], [S4], [S9], [S15] combine two different interactive approaches, which seems to indicate that the authors prefer to focus the interaction on a specific activity or part of the search process. Interestingly, iSBSE studies do not usually delegate the complete evaluation of solutions as originally proposed by IEC, allowing users to influence the search in more flexible ways, such as adding restrictions or selecting solutions.

Second, for the *purpose of user interaction* (see Table 9), a greater proportion (50 percent) of approaches are problem-oriented compared to search-oriented (31 percent). As the former type of interaction is concerned with the application domain, gathering knowledge from the expert seems to be a primary goal when adopting an interactive method. Nevertheless, there are also some primary studies (19 percent) combining both approaches, which allows the decision maker to perform different actions. In fact, the *role of the user* in interaction shows some variety. A more in-depth view reveals that objectives are often adjusted by means of weights [S19], [S8]. However, the addition of constraints [S3], [S4] and providing a subjective evaluation of solutions to complement their fitness value [S1], [S24] are frequent actions when the user acts as enricher. Among search-oriented interactions, there are diverse approaches. When the user is an assistant, the interaction consists in finding promising solutions by marking some of them [S8] or labeling examples [S16]. Guiding the algorithm towards specific regions of the search space often appears in multi-objective approaches [S23], [S7], while tuners are allowed to dynamically change parameters such as the probabilities associated with genetic operators [S9], [S25], [S15].

As shown in Table 10, more than a half of the sources describe evaluation of solutions as the *type of user task*. Selection, comparison and modification of solutions are described in 31, 12 and 23 percent of sources respectively. Such a variety might reflect the emerging nature of this research field, wherein no single approach to user interaction with search is yet predominant. In fact, even though manually fixing solutions could be viewed as a natural action to be performed, modifying solutions mostly appears in combination with other tasks like evaluation and comparison.

When the provided feedback directly or indirectly influences the evaluation phase, the user applies a particular *evaluation mechanism* to achieve this. Table 10 shows that reward/penalization is most frequently occurring at 42 percent of sources, followed by scores (31 percent), and weights (15 percent). Ranking is described in two sources while providing a fitness value is described in one.

TABLE 9
Purpose of Interactivity and Role of the User

Category	Percentage	Papers
Problem-oriented interaction	69%	
Adjuster	35%	[S2], [S8], [S9], [S10], [S14], [S17], [S18], [S19], [S21]
Enricher	35%	[S1], [S3], [S4], [S5], [S6], [S7], [S20], [S24], [S26],
Search-oriented interaction	50%	
Assistant	12%	[S8], [S15], [S16]
Guide	23%	[S7], [S11], [S12], [S13], [S22], [S23]
Tuner	19%	[S1], [S2], [S9], [S15], [S25]

TABLE 10
Tasks Performed During Interaction and Evaluation Mechanisms

Category	Percentage	Papers
Type of task		
Evaluation	58%	[S1], [S2], [S3], [S6], [S8], [S9], [S10], [S12], [S17], [S18], [S19], [S21], [S22], [S23], [S24]
Selection	31%	[S7], [S8], [S11], [S13], [S14], [S15], [S16], [S25]
Comparison	12%	[S4], [S5], [S26]
Modification	23%	[S8], [S9], [S15], [S20], [S23], [S26]
Evaluation mechanism		
Fitness value	4%	[S22]
Weights	15%	[S9], [S17], [S18], [S19]
Scores	31%	[S1], [S6], [S8], [S10], [S11], [S12], [S13], [S24]
Rankings	8%	[S4], [S25]
Reward/penalization	42%	[S2], [S3], [S5], [S13], [S14], [S15], [S16], [S20], [S21], [S23], [S26]
Not applicable	4%	[S7]

Another key aspect of the interactive approach is the *adjustment of interaction time*. Regarding the number of iterations/generations of search between user interaction, the great majority of sources (92 percent) report a fixed number of iterations (see Table 11). Conversely, only two studies apply an adaptive approach, where the time of interaction depends on either how fitness values are progressing [S8] or when a tie is found [S4].

The primary studies provide further detail about the way the user interacts the algorithm, and how solutions are shown to the decision maker. Regarding the *number of solutions* presented at an interactive event, Table 11 shows that 27 percent sources describe the presentation of one solution to the user for evaluation, the remainder describe multiple solutions presented (with the exception of one paper that does not provide any information in this regard). Notice that one significant case of multiple solutions is when the user is requested to pick a pair of solutions from the candidate set in order to compare them [S4], [S5]. The *level of detail* of the presentation to the user is mainly complete detail (at 77 percent), and only four studies [S21], [S26], [S5], [S16] focus the interaction on specific parts of the

TABLE 11
Selection of Solutions and Adjustment of Interaction Time

Category	Percentage	Papers
Adjustment of interaction time		
Fixed	92%	
Every iteration	19%	[S1], [S5], [S10], [S15], [S22]
Every n iterations	35%	[S2], [S6], [S12], [S14], [S17], [S18], [S19], [S24], [S26]
Between two runs	38%	[S3], [S7], [S9], [S11], [S13], [S16], [S20], [S21], [S23], [S25]
Dynamic	8%	
Adaptive	8%	[S4], [S8]
On demand	0%	
Number of solutions		
One solution	27%	[S2], [S3], [S8], [S9], [S10], [S20], [S26]
Pair of solutions	8%	[S4], [S5]
N solutions	38%	[S1], [S6], [S11], [S12], [S13], [S14], [S15], [S19], [S22], [S24]
All solutions	23%	[S7], [S17], [S18], [S21], [S23], [S25]
Not specified	4%	[S16]
Level of detail		
Complete solution	77%	[S1], [S2], [S3], [S4], [S6], [S7], [S8], [S9], [S10], [S11], [S12], [S13], [S14], [S15], [S19], [S20], [S22], [S23], [S24], [S25]
Partial solution	15%	[S5], [S16], [S21], [S26]
Both	8%	[S17], [S18]
Selection strategy		
Fixed	88%	
Best solution(s)	27%	[S2], [S3], [S9], [S19], [S20], [S24], [S26]
Random solution(s)	0%	
Specific solution(s)	27%	[S4], [S5], [S8], [S10], [S11], [S12], [S21]
All solutions	27%	[S1], [S7], [S15], [S17], [S18], [S23], [S25]
Not specified	8%	[S14], [S22]
Free	8%	[S13], [S16]
Not specified	4%	[S6]

TABLE 12
Feedback Integration and Information Lifetime

Category	Percentage	Papers
Feedback integration		
Model-free	81%	[S2], [S3], [S4], [S5], [S6], [S7], [S9], [S11], [S12], [S13], [S14], [S15], [S17], [S18], [S19], [S20], [S21], [S23], [S24], [S25], [S26]
Model-based	19%	[S1], [S8], [S10], [S16], [S22]
Preference information lifetime		
Step-based	50%	[S4], [S6], [S7], [S12], [S13], [S14], [S15], [S17], [S18], [S19], [S21], [S23], [S24]
Short-term	54%	[S1], [S2], [S3], [S5], [S8], [S9], [S10], [S11], [S15], [S16], [S20], [S22], [S25], [S26]
Long-term	0%	
Information validity		
Permanent	54%	[S1], [S4], [S5], [S6], [S9], [S11], [S13], [S14], [S20], [S21], [S22], [S23], [S24], [S26]
Flexible	42%	[S2], [S3], [S7], [S8], [S9], [S10], [S12], [S15], [S17], [S18], [S19]
Unrestricted	15%	[S8], [S15], [S16], [S25]

solutions. A combination of both approaches can be also found in two studies [S18], [S17], where the user is able to get an overview of the candidate solutions in terms of their objective values before entering to inspect the most interesting ones in more detail.

In 88 percent of sources, the *selection strategy* to pick solutions was controlled by the algorithm. It appears that interaction with the user is mostly viewed as a fixed mechanism to reinforce some aspects of the automatic search, instead of an opportunity to create a more flexible, collaborative, user-oriented process. Selecting best, specific or all solutions are equally considered (27 percent). However, random selection of solutions is not described. Some studies propose returning specific solutions by selecting those with the same fitness value so that the user is requested to determine a winner [S4], [S5]. In [S12] a clustering technique is applied to perform the selection. Notice that only one primary source describes the user providing a qualitative fitness value. We suggest that this, combined with a variety of level of solution detail presented, may be an attempt to capture implicit preference information [14] from the user.

Analysis of primary sources, compiled in Table 12, shows that the *feedback integration* within search directly affected the solution models in 19 percent of sources, compared to the great majority (i.e., 81 percent) of sources that are reportedly model-free. Although the latter approaches use ML techniques with different purposes, most of them are focused on modeling the subjective evaluation. For instance, two studies [S8], [S10] propose adjusting the weights associated with the metrics comprising the fitness function by applying a regression model of the user's ratings. Two other studies [S22], [S1] replace user's feedback by using neural networks to build their learning models. In [S1], a regression method (least mean square) is also considered for comparison.

With respect to the *preference information lifetime* of user feedback during search, it is observed that sources describe either a step-based (50 percent) or short-term (54 percent) lifetime. However, long-term lifetimes are not described, that is, information is not reused across different runs of the algorithm to solve similar problems. In addition, with respect to the *information validity* of the user feedback, permanent and flexible account for 54 percent and 42 percent of sources respectively. Notice that these approaches somehow restrict the information flow between the algorithm and the user, who is likely to be interested in ensuring that his/her opinion actually influences the search or changes it in light of new results. Only four sources [S8], [S25], [S16], [S15] allow the user to revoke the decision made during a previous interaction.

5.4 Experimental Framework

Regarding the *type of study* used in experiments, Table 13 indicates that empirical investigations are described in over half of the primary sources (65 percent). Simulated interaction is reported in 35 percent of sources, while sample execution and theoretical proposals account for 4 sources and one source respectively. Table 14 shows the number of participants and their specialization for those primary studies reporting some kind of empirical investigation. With respect to the *number of participants* in experiments, only 8 percent of sources included greater than 20 participants. In fact, acquiring a large number of diverse participants appears to be a major challenge in this area. From the studies including some empirical investigations, 88 percent of sources either partially or completely describe the profession and expertise of the participants.

Analysis of the *participants' position* also reveals that 53 percent of the empirical investigations are conducted with participants who came from an academic background, either as staff (12 percent) or students (35 percent). 35 percent of the empirical studies describe their participants as being software engineering practitioners in an industrial setting. Furthermore, in another four studies [S20], [S23], [S9], [S10], we found other user profiles, although all participants perform the same experimental task. In other cases such as [S20], [S17], [S25], participants are separated into experimental and control groups in order to compare the interactive experience against manually resolving the corresponding task. It is noteworthy that empirical investigations are occasionally conducted in combination with other methods like simulated interactions [S1], [S2], [S18] or sample executions [S9], which provide an efficient comparison framework. In commenting on these findings, we note that the majority of experiments report empirical investigations, providing a necessary level of robustness and rigor associated with experiments involving human behavior.

TABLE 13
Characteristics of the Experiments Conducted in iSBSE

Category	Percentage	Papers
Type of study		
Theoretical proposal	4%	[S7]
Sample execution	15%	[S9], [S11], [S15], [S20]
Simulated interaction	35%	[S1], [S2], [S3], [S4], [S5], [S12], [S18], [S19], [S26]
Empirical investigation	65%	[S1], [S2], [S6], [S8], [S9], [S10], [S13], [S14], [S16], [S17], [S18], [S20], [S21], [S22], [S24], [S25]
Evaluation criteria		
Measures	81%	[S1], [S2], [S3], [S4], [S5], [S6], [S8], [S10], [S11], [S12], [S13], [S14], [S16], [S17], [S18], [S19], [S22], [S23], [S24], [S25], [S26]
Solutions	35%	[S2], [S9], [S11], [S14], [S15], [S16], [S20], [S21], [S23]
Questionnaires	35%	[S1], [S6], [S8], [S10], [S16], [S17], [S18], [S23], [S25]
Case studies		
Synthetic	15%	[S1], [S2], [S9], [S15]
Controlled environment	58%	[S2], [S3], [S8], [S10], [S11], [S14], [S16], [S17], [S20], [S21], [S22], [S23], [S24], [S25], [S26]
Industrial case	38%	[S1], [S4], [S5], [S6], [S13], [S18], [S19], [S20], [S23], [S26]
Not specified	4%	[S12]
Evidence		
Source code	12%	[S1], [S5], [S20]
Case studies	27%	[S1], [S3], [S5], [S10], [S11], [S13], [S20]
Experimental results	8%	[S5], [S10]
Solutions	0%	
Questionnaires	0%	
Transcripts	12%	[S11], [S13], [S20]
Additional statistics	4%	[S1]
Other	8%	[S5], [S10]
Statistical tests		
Yes	58%	
Pairwise comparison	50%	[S1], [S3], [S6], [S8], [S10], [S12], [S17], [S18], [S20], [S22], [S23], [S25], [S26]
Multiple comparison	19%	[S4], [S5], [S6], [S10], [S12]
Effect size measurement	12%	[S1], [S17], [S26]
No	42%	[S2], [S7], [S9], [S11], [S13], [S14], [S15], [S16], [S19], [S21], [S24]

The *evaluation criteria* determine which mechanism is used to analyze the experimental outcomes. In interactive search experiments (see Table 13), the majority of the analyzed sources (81 percent) report the use of measures, i.e., fitness values or specific assessment metrics. With respect to the scope of the software artifacts deployed as *case studies* in experiments, 58 percent of sources report controlled environments. On the other hand, industrial case studies are reported in 38 percent of sources, and four sources describe the use of synthetic case studies. We note that, while acknowledging the logistical challenges involved, it is clear that from the point of view of iSBSE as a requisite and precursor to the adoption of industrial SBSE tools, experiments involving industrial case studies are to be preferred over controlled environments and synthetic simulations.

It is remarkable that only some studies are accompanied with *evidence* in the form of additional experimental material. Table 13 reflects the nature of the reachable materials, which were found in 27 percent of sources. Finally, we find that 58 percent of studies employ *statistical tests* to validate experimental results. Pairwise and/or multiple comparison techniques are reported in 15 sources, while estimation of effect size is conducted in 3 (12 percent). Table 15 shows the details of the specific statistical tests applied in each case study. As can be observed, the Wilcoxon Mann-Whitney test is preferred for pairwise comparisons. There is less agreement for multiple comparison or effect size measurement, and their use is not that usual either. In commenting on these findings, we would encourage authors to make experimental materials available for readers. Furthermore, in recognition of the stochastic nature of metaheuristic search and the variability of human behavior, we would also encourage authors to always make use of appropriate statistical analysis.

6 Cross Category Data Analysis

As a response to RQ3, in this section we review the extracted data across categories of the classification scheme as a way to identify potential gaps and limitations of iSBSE, while speculating about possible causes. We first compare and contrast extracted data relating to the software engineering *problem formulation* with data in the various categories of *search technique* and *interactive approach* used in the primary sources. Second, we contrast the data regarding *search techniques* in the primary sources with the data in various categories of *interactive approach*. To illustrate possible associative relationships, bubble charts revealing the frequency of occurrence of data from various categories are shown. Note that statistical analysis was considered as a means to determine the presence of correlation. However, due to sparsity and a lack of independence of the categorical data, this proved inappropriate.

TABLE 14
Characteristics of the Participants in Empirical Studies (User's Profile)

Number of participants	
1 participant	
Between 2 and 10	[S1], [S9], [S10], [S13], [S14], [S16], [S18], [S21], [S24]
Between 11 and 20	[S2], [S8], [S20], [S22], [S23], [S25]
More than 20	[S6], [S17]
Participant position	
Engineer (industry)	[S1], [S2], [S13], [S18], [S22], [S23]
PhD academia staff	[S9], [S10], [S22]
Postgraduate student	[S9], [S17], [S20], [S21], [S22], [S23], [S24]
Undergraduate student	[S10], [S20], [S22], [S25]
Other	[S6], [S8]
Not specified	[S14], [S16]
Participant expertise (in years)	
Less than 5	[S2], [S10], [S17], [S18], [S22], [S23]
Between 5 and 10	[S1], [S2], [S10], [S13], [S18], [S22], [S23]
Between 11 and 20	[S2], [S10], [S13], [S18], [S22], [S23]
More than 21	[S10]
Not specified	[S6], [S8], [S9], [S20], [S21], [S24], [S25]

6.1 Problem Formulation and Approaches

Fig. 2a reveals the frequencies of search techniques used in various problem formulations. As can be seen, evolutionary computation is widely adopted in almost all SE phases, but is frequently used in analysis and design (9 studies) and maintenance (6 studies) problems. It is worth noting that other metaheuristics like swarm intelligence [S8] and hill climbing [S20] have been also applied very recently for the first time in the area. For instance, first results on the application of swarm intelligence for requirement elicitation were published in 2016 [S2].

Fig. 2b shows how a great variety of types of interactive algorithms have been already employed to address analysis and design problems. We observe that iSBSE was first used in design tasks. As a result, the landscape of approaches in this area is more extensive, including the only two primary studies on human-guided search [S9], [S15]. An interesting finding here is that, even though the automatic evaluation of solutions is a well-known challenge in SBSE, it remains a difficult task for the software engineer as 'human-in-the-loop', and so indirect mechanisms to evaluate candidate solutions are preferred in iSBSE. This is reflected by the fact that PI (preference-based interactivity) and IR (interactive optimization) are more frequently used than HE (human-based evaluation) in all SE areas. As for PI, it is the most frequent interactive approach for the design and maintenance categories of problem formulation, meaning that researchers find it appropriate to leverage the designer's abilities for recognizing promising solutions. It is noteworthy that this approach is not used for requirements and testing problems, wherein human preferences often abound.

Fig. 2c reveals that EV (evaluation) is frequently used across a range of problem formulations (with the exception of source code development). This might reflect a relative ease of implementation by simply presenting candidate solution(s) in a visualization for either direct or indirect evaluation, when compared with other interactive tasks. An interest in gathering information to reinforce the problem formulation predominates. It is also noticeable that CO (comparison) is not found in relation to analysis and design. Apparently, this task might be well suited to design problems, though the implementation difficulties caused by visually presenting multiple complete solutions might represent the major limitation at this point.

TABLE 15
Statistical Tests Applied in iSBSE

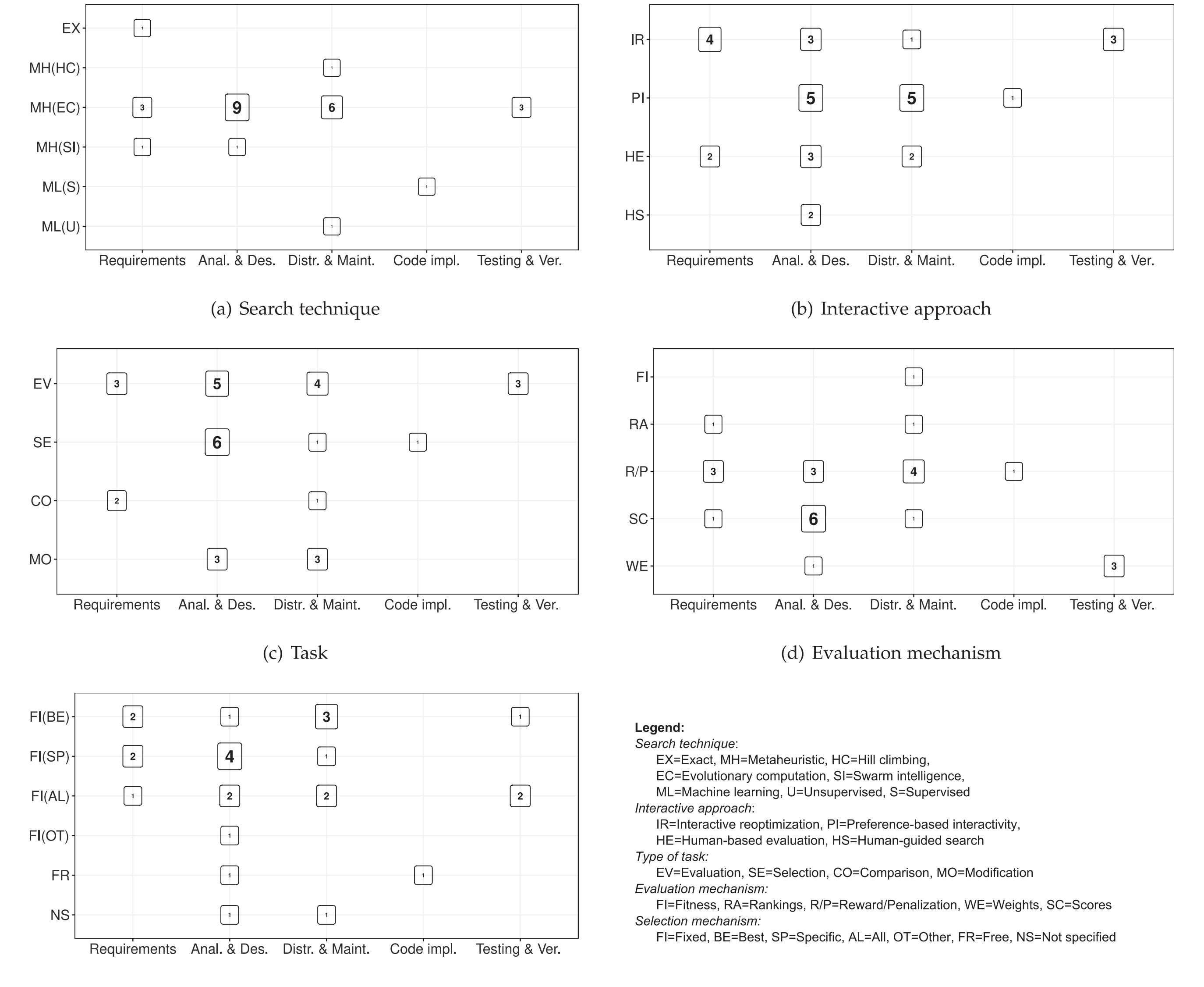
Pairwise comparison	
Student's t-test	[S8], [S25]
Wilcoxon Mann-Whitney	[S1], [S3], [S6], [S17], [S18], [S20], [S22] [S8], [S10], [S12], [S23], [S26]
Multiple comparison	
ANOVA	[S4], [S5]
Friedman	[S10]
Kruskal-Wallis	[S6], [S12]
Effect size measurement	
Cliff's Delta	[S1], [S17]
Vargha and Delaney	[S26]

Fig. 2d illustrates the frequencies of interactive evaluation mechanisms used across various problem formulations. We can observe here that evaluation based on RA (rankings) is the least frequently occurring, and is confined to requirements and maintenance problems. It is interesting to speculate that this may in some part be due to the relative difficulty of implementing a RA mechanism as part of the visualization of multiple candidate solutions for an interactive user experience. To overcome such a limitation, the amount of information to be shown needs to be reduced and somehow limited to a subset of solutions. For example, the sorting procedure applied to requirements prioritization [S4] only displays those solutions showing a conflict in the priority order within a pair of requirements. In addition, the information is limited by setting a maximum number of disagreements to be solved.

Focusing on selection mechanisms for the presentation of candidate solutions (see Fig. 2e), it is apparent that showing a fixed numbers of solutions—especially BE (best) and AL (all) solutions—is preferred in almost all problem domains. Showing specific solutions might be of highest interest to the user, i.e., FI(SP) (fixed, specific). It was considered in the earliest stages of iSBSE to face requirements or design problems [S12], [S11], [S5], [S10]. However, in the last few years studies seem to be more oriented towards BE and AL, i.e., selecting the best solution [S2], [S20] or all of them [S1], [S17], independently of the problem domain.

Taking the extracted data results analysis shown in Figs. 2a to 2e overall, it is clear that some categories of search types and interactive approaches do occur for some problem formulations in the primary studies more frequently than others. To summarize and highlight these with respect to the following problem formulations:

- *Requirements*: *Interactive reoptimization* is the most frequently occurring interactive algorithm, and with



(a) Search technique

(b) Interactive approach

(c) Task

(d) Evaluation mechanism

(e) Selection mechanism

Fig. 2. Relation between software engineering phases and other categories.

reward/penalization the most frequent problem evaluation mechanism.

- *Design*: *Preference-based interactivity* is the most frequently occurring interactive algorithm, with *selection* and *evaluation* the most frequent interactive tasks for the user, together with *scores* as the the most frequent solution evaluation mechanism.
- *Maintenance*: *Preference-based interactivity* is again the most frequently occurring interactive algorithm, and *reward/penalization* the most frequent solution evaluation technique.

For code and testing, because of the paucity of studies, it seems too early to reach meaningful conclusions and make suggestions about how interactivity is being considered within these fields.

6.2 Search Technique and Interactive Approach

Focusing on the search algorithms, there is no clear trend regarding the application of a specific interactive model. Some recent studies apply human-based evaluation [S1], [S6]. Other approaches like preference-based interactivity and interactive reoptimization can be also found over the years, the latter being applied by an evolutionary algorithm for the first time in 2012 [S10]. Likewise, ant colony optimization was first applied following the precepts of human-based evaluation [S8], whereas interactive reoptimization has been considered more recently in 2016 [S2]. In addition, we note that machine learning does not directly work on candidate solutions as SBSE does, and the problem is not defined in terms of a fitness function. Consequently, the studies within the ML paradigm fall into the same interactive approach (PI).

By a wide margin, the most frequently occurring interactive task is for evaluation of solutions and evolutionary computation (with 13 studies). Notice that the ultimate goal of user interaction is to provide some kind of evaluation, at least indirectly. This can be observed for swarm intelligence too, given that the two studies using the ACO paradigm [S2], [S8] focus their interaction on the evaluation phase. However, [S8] combines this task with the selection and modification of solutions. The latter also appears in combination with evaluation in three evolutionary-based studies [S23], [S9], [S26], which suggests that this decision is mostly motivated by the specific requirements of the problem domain. It is worth noting that both primary studies based on machine learning [S25], [S16] propose that the human participates in selecting input data to enable the subsequent mining process to produce more accurate classification models. However, how the user could help to improve other phases of the learning process (such as selecting features of interest or interactively supervising the construction of the classifier) remains as yet unexplored.

In analyzing how frequently each search technique applies an evaluation or a selection mechanism, little or no correspondence is immediately apparent. More specifically, it seems that there exists an independence between the underlying search technique (acting as a ’search engine’) and the implementation of the user’s interactive experience, at least in terms of interactive task, evaluation mechanism, and selection of candidate solutions for presentation and visualization. This is a valuable aspect of iSBSE, since it offers the flexibility to incorporate a customized interactive experience related to the characteristics of the problem domain independently from the underlying ’search engine’.

7 Open Issues and Future Trends

Taking the results of the systematic literature review presented in previous sections overall, it is evident that iSBSE has attracted significant research attention over recent years and substantive progress has been made. However, the review also reveals a number of challenges. In the following, we comment on some open issues and also speculate on future trends, as a response to RQ4.

Distance between the ’Optimizer’ and the Software Engineer. In a real industrial setting, most probably those researchers who design and develop iSBSE software systems may not be the same persons acting as SE stakeholders to provide the domain-specific requirements, and subsequently use and interact with the iSBSE software system. If so, several issues might arise. For example, iSBSE requirements elicitation and analysis is crucial particularly with regard to the required interactive user experience. In addition, search algorithms chosen for implementation may not always be entirely appropriate for the given software engineering problem domain and development process. We speculate that it would be highly advantageous to engage with SE stakeholders, possibly by means of pilot studies, to validate and refine the user interactive experience, and its relationship to the search ’optimizer’, prior to conducting experimental investigations.

User Trust and Acceptance. Achieving user trust and acceptance remains an open challenge for the field of iSBSE. The issue of ’automation bias’ [15] is highly relevant, as is user complacency [16], [17]. Lyell and Coiera [15] suggest that strategies to minimize automation bias might focus on cognitive load reduction. The suggestion is echoed by Jackson [56] who points out that supporting human understanding is an important role for automation in software engineering, and it seems possible that wherever there is a lack of understanding, a lack of user trust and acceptance might follow. In addressing this challenge, it is noteworthy that Marculescu et al. [S17] report the use of the NASA Task Load Index (NASA-TLX), a subjective workload assessment tool (see Hart and Staveland [57]). It is also noteworthy that qualitative methods such as Grounded Theory [58] may have a role to play in the analysis of free format questionnaire data. Unlike quantitative approaches, where typically a pre-formed theory is proposed (possibly as one or more hypotheses) and experiments are designed to collect data to prove or refute hypotheses, Grounded Theory starts without a pre-formed theory and through repeated data collection and analysis, allows a theory to emerge from the data [59], [60]. As part of efforts to achieve user trust and acceptance in iSBSE, we would encourage consideration of

- techniques such as cognitive load monitoring and qualitative methods to examine the interactive user experience, and
- effective mechanisms to hide the underlying complexity of search techniques, such as reducing the number of parameters required of the user.

Role of the User. The nature of user participation in iSBSE also remains an open issue for the field. As observed in Section 5.3, it is possible that less rigid interaction mechanisms beyond evaluation of solutions, including direct manipulation of candidate solutions and the search process itself, may prove beneficial. However, previous work from other fields may offer possibilities to address this. For example, Kosorukoff [61] proposes a human-based genetic algorithm, wherein all primary genetic operators are delegated to the human. Kosorukoff asserts that the advantage of such an approach is its ability to not only facilitate the direct evaluation of individuals, but also to identify a good representation for them. In a further example, Bush and Sayama [62] propose hyperinteractive evolutionary computation (HIEC), in which the user pro-actively chooses when and how each evolutionary operator is applied. In human-subject experiments comparing HIEC with interactive evolutionary computing, Bush and Sayama report that HIEC facilitated a *“more controllable, more fun and more satisfying”* user experience. In a more recent study, Byrne et al. [63] propose an interactive methodology to enable the user to directly interact with the encoding of the candidate solutions they find aesthetically pleasing, and assert that *“broadening interaction beyond simple evaluation increases the amount of feedback and bias a user can apply to the search.”* Given the crucial role of the user in iSBSE, we would also encourage a flexible and rich role for the user, possibly by drawing inspiration from the wider interactive metaheuristic search research community.

Assessment of Solution Fitness. User evaluation of solution fitness remains an open challenge to the iSBSE community, as software engineers can struggle to bring their ’hidden’ knowledge to bear during interaction. ’Hidden’ knowledge can relate to user qualitative evaluation of aesthetic aspects, such as the visual appeal of graphical user interfaces, or software design elegance. For example, Quiroz et al. [S14] compute the fitness of a graphical user interface using a weighted combination of user aesthetic evaluation and interface design guidelines. Troiano et al. [S6] search for a good compromise between interface aesthetics and accessible color schemes, taking account of two different fitness factors: an objective measure of color distances and contrast ratios found in the user interface and a subjective aesthetic score given by user. In a further example of the use of ’hidden’ knowledge [S10], user measures relating to aspects of software design elegance (e.g., symmetry) are integrated with more established measures of coupling and cohesion during search to achieve more 'human-looking' solutions. Moreover, the notion of distinguishing user preferences as either hidden or implicit (as opposed to explicit) is explored by Aljawadeh et al. [14]. Results of this review suggest that user preferences can be short term or long term, which is consistent with the concept of a dynamic and flexible user experience recommended by Aleti et al. [26]. In addition, user fatigue is a crucial factor for the assessment of solution fitness [2]. Some tactics to reduce its effects have been offered by Shackelford [19] and these have been extended in the 'interactive solution presentation' metaheuristic design pattern [64]. Furthermore, reflecting the SBSE-related trend to Pareto-optimal many/multi-objective approaches [30], human solution fitness assessment and dynamic trade-off analysis present a challenge for many/multi-objective search algorithms. Current approaches are mostly focused on the dynamic adjustment of objectives during search [S17], [S9], which simplifies the problem into a single-objective one. Only a few studies maintain the original problem formulation, trying to infer user's judgment from actions such as the prioritization of non-dominated solutions [S23] or by means of sub-populations whose size varies with the relative importance of each objective as perceived by the user [S10]. It is interesting to observe that although many reports comparing multi/many objective Pareto-optimal approaches have appeared in the SBSE field [65], [66], [67], reports of dynamic, "in-the-loop" adjustment of the relative user importance of objective trade-offs for these approaches are less readily available, and we speculate that this might seem a fruitful avenue for future research.

Choice of Search Technique. As revealed in Section 5.2, evolutionary computation has been chosen as a search technique across a variety of SE problem formulations. However, it is interesting to note that studies comparing the relative performance of various metaheuristics generally, and iSBSE specifically, are not abundant in the literature, especially with regard to multi-objective optimization. Having said that, in a recent study, Piotrowski et al. [68] compare the relative performance of 33 metaheuristic techniques on 22 numerical real-world problems across a range of scientific domains. Results obtained would appear to be broadly consistent with those previously observed by Simons and Smith [69], who conducted a study comparing the performance of greedy local search, an evolutionary algorithm and ACO for search-based software design. Piotrowski et al. report that *"Particle Swarm Optimization algorithms and some new types of metaheuristics perform relatively better when the number of allowed function calls is low, whereas Differential Evolution and Genetic Algorithms perform better relative to other algorithms when the computational budget is large*". Such findings resonate with the need to achieve effective search performance quickly in iSBSE, wherein reducing waiting times might help to mitigate the sensation of fatigue perceived by the user. While the findings of these two studies cannot be considered to be a comprehensive and complete picture of this open issue, they do however suggest that the use of swarm intelligence is worthy of consideration when choosing a search technique for iSBSE, particularly for multi-objective scenarios.

Available Frameworks and Implementations. Following the choice of search technique, a further open issue relates to the extent to which the availability of development frameworks for metaheuristic search might influence the choice of the search technique. Mature frameworks for the development of evolutionary computation approaches are readily available (e.g., jMetal [70], JCLEC [71]) and it is possible that this availability might contribute to the bias towards evolutionary algorithms found in the review results. It also seems possible that iSBSE researchers might not necessarily have appropriate expertise in optimization or search algorithms, and so seek to build on advances in the field by utilizing benchmarked implementations of algorithms. It is interesting to speculate that comparable studies (e.g., of similar algorithm implementations) could be a criterion for researchers in their choice of a search technique.

Experimental Validity and Empirical Studies. As discussed in Section 5.4, experimental evaluation of iSBSE research with respect to both research and empirical industrial contexts remains an on-going challenge. Given the emerging state of the iSBSE field, there appears to be a general lack of readily agreed and available benchmark case studies and standards. Considering the presence of human interaction with search, statistical analysis is appropriate, although it is recognized that this might be difficult if the sample size (in terms of number of participants and problem instances) is small. To address this, we note that Arcuri and Briand offer a helpful guide to statistical tests for assessing randomized algorithms in software engineering [47], and Neumann et al. suggest a useful executable experimental template pattern for the systematic comparison of metaheuristics [72]. Furthermore, important aspects like reproducibility are also related to additional materials being made available, allowing experiments and results to be contrasted. However, the results of this review reveal that additional experimental materials are made available in just 27 percent of the primary studies, and we would encourage authors to consider this, respecting ethical and confidentiality issues where appropriate. Notice that the evaluation of interactive sessions is essential to give insights about the effectiveness of the interaction with the software engineer. However, the lack of commonly accepted guidelines to design and conduct experiments in iSBSE makes difficult the comparison of different proposals. In this context, the use of common problem instances and mechanisms to collect and process data, as well as unifying the creation of questionnaires would be highly recommended. It also seems likely that, with regard to industrial case studies in iSBSE, it remains a logistic challenge to recruit sufficient participants of sufficient expertise to be representative of some subset of the software engineering practitioner community. Lastly, in the longer term, we suggest that assessing the economic benefit of iSBSE (as suggested by Aleti et al. [26] for the field of Operations Research), may be important for promoting future industrial adoption.

Combining other Artificial Intelligence Techniques with iSBSE. We also speculate that one future trend lies in the combination of other artificial intelligence techniques, e.g., machine learning, with iSBSE, to learn implicit user preferences and thus help reduce user fatigue and improve the interactive user experience. It is interesting to note that in one of the primary studies, in addressing the next release problem, Araújo et al. [S1] report that the expertise of the decision maker is more effectively incorporated when machine learning techniques integrate tacit human knowledge with the optimization process. A further example of research combines machine learning and SBSE, where Amal et al. [S22] report the application of machine learning to classify ill-defined, implicit fitness functions using a case study on search based software refactoring. In a different (non-interactive) approach, Aleti et al. [73] incorporate a probabilistic Bayesian heuristic during search-based optimization of component deployment in embedded software systems. This is achieved by calculating the *a posteriori* probability that a particular component/host assignment constitutes a good outcome, which is then used to identify high quality deployed architectures, given a number of observations during search. These examples suggest that learning from the optimization process can provide valuable knowledge in SBSE, but further research is required into how reuse of such knowledge might be exploited when addressing related problems. We speculate that future research in iSBSE might investigate the opportunities that artificial intelligence can bring beyond the partial substitution of the human evaluation.

After an in-depth analysis of iSBSE, we have also observed some *gaps in the field* that might turn into research opportunities. As revealed in Section 5.1, results of the review indicate that *some lifecycle project phases appear to be as yet unexplored* in the primary studies. The lack of iSBSE studies in the areas of source code implementation and project management would seem to indicate a gap in the field. As noted previously, the deficiency of research studies into interactive search-based project management is interesting, considering its analogy to requirements prioritization, and the fact that others have investigated interactive search-based project management scheduling approaches more generally [19], [74]. In addition, we speculate that *incorporating expert knowledge into search* in iSBSE represents a gap in the field. As is typical in metaheuristic search, sources in the review initialize populations of solution individuals either at random or according to domain-specific heuristics. However, the reuse of expert knowledge (e.g., patterns, benchmarks etc.) to initialize populations might offer possibilities. We also observe that while individual phases of development are well supported by iSBSE, each is presented as if isolated from others, with different solution representations and search operators. Knowledge may be generated by search at one phase, but often discarded thereafter. We conjecture that a fruitful focus of research might address the *bringing together these isolated search elements in a development chain*, aligned to the software engineering development process.

8 Threats to Validity

Threats to the validity of this systematic review mainly relate to factors that might affect outcomes without the researchers' knowledge. Such threats center principally on the selection of primary studies and the process of extracting the appropriate data from the primary studies.

Regarding the selection of primary studies for the review, we describe in Section 3.1 the steps taken to justify their systematic and unbiased selection. To ensure relevant studies were located, we formulated a list of search strings derived from terms from a variety of application domains and search techniques. We conducted multiple iterative pilot searches to validate the adequacy and effectiveness of the search strings, looking for publications in the broadest sense. When required, a search string was adapted to the specific query language used by a particular engine or database. A yield of 653 sources returned suggests a level of confidence in the breadth of search. Having identified relevant sources using the search strings, we drafted and composed sets of inclusion and exclusion criteria to specifically identify those sources related to interactive SBSE. In Section 3.2, we state the inclusion and exclusion criteria to precisely describe the scope of the interactive role of the software engineer in SBSE. After applying inclusion and exclusion criteria, a resulting focus on 26 remaining primary sources out of the 669 originally returned by search also suggests a level of confidence in the inclusion and exclusion process.

With regard to the process of accurate and objective data extraction from the primary sources, we first formulated an appropriate classification scheme (see Section 3.4), which precisely describes the data items to be extracted. The classification scheme was derived by reference to other rigorous classifications of interactive metaheuristic search previously presented in other fields, and the specific context of SBSE. Second, each primary source was independently scrutinized by at least two reviewers, and in the case of any disagreements occurring, a third reviewer's assessment was brought to bear as arbiter. To prevent any potential conflict of interest, reviewers did not participate in data extraction where they were also an author of the primary source. Finally, we believe that the data extraction review process itself provided a thorough and robust validation of the classification scheme, which we hope may itself prove a useful contribution and starting point for other researchers in the field.

9 Conclusion

The design, development and maintenance of complex software systems can be challenging for the software engineer to perform. The application of computational optimization methods has been shown to be an efficient mechanism to support the engineer in different software engineering tasks during development. However, there are still many problems where human intervention is crucial for solution evaluation and indication of preferences to the algorithm about how a given situation should be solved. In this work, we present the results of a systematic literature review into interaction within search-based software engineering. Such interaction occurs when the software engineer provides their implicit and/or explicit feedback during the search process, which significantly modifies the results and performance of the search.

To enable analysis of the primary studies, we formulate a classification scheme focused on four key aspects of iSBSE, i.e., the problem formulation, search technique, interactive approach, and the empirical framework used in experiments. Using this classification scheme, we analyzed the primary sources and present the results in manner that enables researchers to relate their work to the current body of knowledge and recognize future trends. We organized our results into tables and graphics, aiming to enable knowledge transfer for industrial practitioners, and across research communities whose focus may lie in distinct phases across the software development lifecycle. We note that research into iSBSE is multi-disciplinary in the sense of drawing on three areas, i.e., human-computer interaction, artificial intelligence (e.g., metaheuristic search, machine learning etc.), and software engineering. It seems unlikely that researchers might possess expertise in all three areas, and so this review helps to bridge the gap between the discipline-based communities and promote inter-disciplinary knowledge transfer.

The review and analysis based on the classification scheme highlights some open research issues for the field of iSBSE. The role of the user during interaction with search remains a challenge for researchers, particularly with regard to provision of a flexible and dynamically adapting user experience to promote user trust and acceptance of the search results obtained. We note that this is non-trivial to implement, and so conclude that it is crucial to take account of user requirements, software engineering context and human factors in the development of an iSBSE system. The generation and management of knowledge during iSBSE also represents an interesting and fruitful research trend for iSBSE. Currently, much knowledge about the problem formulation and candidate solutions is generated during interactive search, but is seldom retained for longer term use. We also conclude that future research might fruitfully be directed towards the acquisition and exploitation of knowledge generated during interactive search.

Finally, the results of this systematic review will enable the advancement of the iSBSE research community as the area continues to grow, and hopefully the classification scheme presented herein might form the beginning of a methodological approach for the design of iSBSE systems.

ACKNOWLEDGMENTS

This work is supported in part by the Spanish Ministry of Economy and Competitiveness, projects TIN2017-83445-P and TIN2015-71841-REDT, the Spanish Ministry of Education under the FPU program (FPU13/01466), and FEDER funds.

PRIMARY STUDIES

[S1] A. A. Araújo, M. Paixao, I. Yeltsin, A. Dantas, and J. Souza, "An Architecture based on interactive optimization and machine learning applied to the next release problem," *Autom. Softw. Eng.*, vol. 24, no. 3, pp. 623–671, 2017.

[S2] T. do Nascimento Ferreira, A. A. Araújo, A. D. B. Neto, and J. T. de Souza, "Incorporating user preferences in ant colony optimization for the next release problem," *Appl. Softw. Comput.*, vol. 49, pp. 1283–1296, 2016.

[S3] A. Dantas, I. Yeltsin, A. A. Araújo, and J. Souza, "Interactive software release planning with preferences base," in *Proc. 7th Int. Symp. Search-Based Softw. Eng.*, 2015, pp. 341–346.

[S4] P. Tonella, A. Susi, and F. Palma, "Interactive requirements prioritization using a genetic algorithm," *Inf. Softw. Technol.*, vol. 55, no. 1, pp. 173–187, 2013.

[S5] F. Palma, A. Susi, and P. Tonella, "Using an SMT Solver for Interactive Requirements Prioritization," in *Proc. 19th ACM SIGSOFT Symp. / 13th Eur. Conf. Found. Softw. Eng.*, 2011, pp. 48–58.

[S6] L. Troiano, C. Birtolo, and R. Armenise, "A validation study regarding a generative approach in choosing appropriate colors for impaired users," *SpringerPlus*, vol. 5, no. 1, 2016, Art. no. 1090.

[S7] A. E. El Yamany, M. Shaheen, and A. S. Sayyad, "OPTI-SELECT: An interactive tool for user-in-the-loop feature selection in software product lines," in *Proc. 18th Int. Softw. Product Line Conf.: Companion Vol. Workshops, Demonstrations Tools*, 2014, vol. 2, pp. 126–129.

[S8] C. L. Simons, J. Smith, and P. White, "Interactive ant colony optimization (iACO) for early lifecycle software design," *Swarm Intell.*, vol. 8, no. 2, pp. 139–157, 2014.

[S9] S. Vathsavayi, H. Hadaytullah, and K. Koskimies, "Interleaving human and search-based software architecture design," *Proc. Estonian Academy Sci.*, vol. 62, no. 1, pp. 16–26, 2013.

[S10] C. L. Simons and I. C. Parmee, "Elegant object-oriented software design via interactive, evolutionary computation," *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, vol. 42, no. 6, pp. 1797–1805, Nov. 2012.

[S11] C. L. Simons, I. C. Parmee, and R. Gwynllyw, "Interactive, evolutionary search in upstream object-oriented class design," *IEEE Trans. Softw. Eng.*, vol. 36, no. 6, pp. 798–816, Nov./Dec. 2010.

[S12] C. Birtolo, P. Pagano, and L. Troiano, "Evolving colors in user interfaces by interactive genetic algorithm," in *Proc. World Congress Nature Biol. Inspired Comput.*, 2009, pp. 349–355.

[S13] C. L. Simons and I. C. Parmee, "An empirical investigation of search-based computational support for conceptual software engineering design," in *Proc. IEEE Int. Conf. Syst. Man Cybern.*, 2009, pp. 2503–2508.

[S14] J. C. Quiroz, S. J. Louis, A. Shankar, and S. M. Dascalu, "Interactive genetic algorithms for user interface design," in *Proc. IEEE Congress Evol. Computation*, 2007, pp. 1366–1373.

[S15] N. Monmarché, G. Nocent, M. Slimane, G. Venturini, and P. Santini, "Imagine: A tool for generating html style sheets with an interactive genetic algorithm based on genes frequencies," in *Proc. IEEE Int. Conf. Syst. Man Cybern.*, 1999, vol. 3, pp. 640–645.

[S16] S. Axelsson, D. Baca, R. Feldt, D. Sidlauskas, and D. Kacan, "Detecting defects with an interactive code review tool based on visualisation and machine learning," in *Proc. 21st Int. Conf. Softw. Eng. Knowl. Eng.*, 2009, pp. 412–417.

[S17] B. Marculescu, S. Poulding, R. Feldt, K. Petersen, and R. Torkar, "Tester interactivity makes a difference in search-based software testing: A controlled experiment," *Inf. Softw. Technol.*, vol. 78, pp. 66–82, 2016.

[S18] B. Marculescu, R. Feldt, R. Torkar, and S. Poulding, "An initial industrial evaluation of interactive search-based testing for embedded software," *Appl. Softw. Comput.*, vol. 29, pp. 26–39, 2015.

[S19] B. Marculescu, R. Feldt, and R. Torkar, "Objective Re-weighting to guide an interactive search based software testing system," in *Proc. 12th Int. Conf. Mach. Learn. Appl.*, Dec. 2013, vol. 2, pp. 102–107.

[S20] Y. Lin, X. Peng, Y. Cai, D. Dig, D. Zheng, and W. Zhao, "Interactive and guided architectural refactoring with search-based recommendation," in *Proc. 24th ACM SIGSOFT Int. Symp. Found. Softw. Eng.*, 2016, pp. 535–546.

[S21] M. W. Mkaouer, "Interactive code smells detection: An initial investigation," in *Proc. 8th Int. Symp. Search-Based Softw. Eng.*, 2016, pp. 281–287.

[S22] B. Amal, M. Kessentini, S. Bechikh, J. Dea, and L. B. Said, "On the use of machine learning and search-based software engineering for Ill-defined fitness function: A case study on software refactoring," in *Proc. 6th Int. Symp. Search Based Softw. Eng.*, 2014, pp. 31–45.

[S23] M. W. Mkaouer, M. Kessentini, S. Bechikh, K. Deb, and M. Ó Cinnéide, "Recommendation system for software refactoring using innovization and interactive dynamic optimization," in *Proc. 29th ACM/IEEE Int. Conf. Automat. Softw. Eng.*, 2014, pp. 331–336.

[S24] A. Ghannem, G. El Boussaidi, and M. Kessentini, "Model refactoring using interactive genetic algorithm," in *Proc. 5th Int. Symp. Search Based Softw. Eng.*, 2013, pp. 96–110.

[S25] J. Wang, X. Peng, Z. Xing, and W. Zhao, "Improving feature location practice with multi-faceted interactive exploration," in *Proc. Int. Conf. Softw. Eng.*, 2013, pp. 762–771.

[S26] G. Bavota, F. Carnevale, A. De Lucia, M. Di Penta, and R. Oliveto, "Putting the developer in-the-loop: An interactive GA for software re-modularization," in *Proc. 4th Int. Symp. Search Based Softw. Eng.*, 2012, pp. 75–89.

REFERENCES

[1] M. Harman and B. F. Jones, "Search-based software engineering," *Inf. Softw. Technol.*, vol. 43, no. 14, pp. 833–839, 2001.

[2] M. Harman, S. A. Mansouri, and Y. Zhang, "Search-based software engineering: Trends, techniques and applications," *ACM Comput. Surv.*, vol. 45, no. 1, 2012, Art. no. 11.

[3] D. Meignan, S. Knust, J.-M. Frayret, G. Pesant, and N. Gaud, "A review and taxonomy of interactive optimization methods in operations research," *ACM Trans. Interact. Intell. Syst.*, vol. 5, no. 3, 2015, Art. no. 17.

[4] B. Curtis, H. Krasner, and N. Iscoe, “A field study of the software design process for large systems,” *Commun. ACM*, vol. 31, no. 11, pp. 1268–1287, 1988.

[5] R. Guindon, “Designing the design process: Exploiting opportunistic thoughts,” *Human-Comput. Interaction.*, vol. 5, no. 2, pp. 305–344, 1990.

[6] A. E. Eiben and J. E. Smith, *Introduction to Evolutionary Computing*. Berlin, Germany: Springer, 2015.

[7] G. Klien, D. D. Woods, J. M. Bradshaw, R. R. Hoffman, and P. J. Feltovich, “Ten challenges for making automation a ”team player” in joint human-agent activity,” *IEEE Intell. Syst.*, vol. 19, no. 6, pp. 91–95, Nov./Dec. 2004.

[8] D. Jones, “Programming using genetic algorithms: isn’t that what humans already do.” [Online]. Available: http://shape-of-code.coding-guidelines.com/2013/10/18/programming-using-genetic-algorithms-isnt-that-what-humans-already-do/, Accessed on: 8-Nov.-2016

[9] C. Le Goues, S. Forrest, and W. Weimer, “Current challenges in automatic software repair,” *Softw. Quality J.*, vol. 21, no. 3, pp. 421–443, 2013.

[10] R. Dawkins, *The Blind Watchmaker*. Harlow, U.K: Longman, 1986.

[11] K. Sims, “Evolving virtual creatures,” in *Proc. 21st Ann. Conf. Comput. Graph. Interactive Tech.*, 1994, pp. 15–22.

[12] H. Takagi, “Interactive evolutionary computation: Fusion of the capabilities of EC optimization and human evaluation,” *Proc. IEEE*, vol. 89, no. 9, pp. 1275–1296, Sep. 2001.

[13] J. Branke, K. Deb, K. Miettinen, and R. Słowiński, *Multiobjective Optimization: Interactive and Evolutionary Approaches*. Berlin, Germany: Springer, 2008, vol. LCNS 5252.

[14] H. J. Aljawawdeh, C. L. Simons, and M. Odeh, “Metaheuristic design pattern: Preference,” in *Proc. Companion Publication Ann. Conf. Genetic Evol. Comput.*, 2015, pp. 1257–1260.

[15] D. Lyell and E. Coiera, “Automation bias and verification complexity: A systematic review,” *J. Am. Med. Inform. Assoc.*, vol. 24, 2016, Art. no. ocw105.

[16] R. Parasuraman and V. Riley, “Humans and automation: Use, misuse, disuse, abuse,” *Human Factors: J. Human Factors Ergonomics. Soc.*, vol. 39, no. 2, pp. 230–253, 1997.

[17] L. J. Skitka, K. L. Mosier, and M. Burdick, “Does automation bias decision-making?” *Int. J. Human.-Comput. Stud.*, vol. 51, no. 5, pp. 991–1006, 1999.

[18] J. E. Bahner, A.-D. Hüper, and D. Manzey, “Misuse of automated decision aids: Complacency, automation bias and the impact of training experience,” *Int. J. Human.-Comput. Stud.*, vol. 66, no. 9, pp. 688–699, 2008.

[19] M. Shackelford, “Implementation issues for an interactive evolutionary computation system,” in *Proc. Companion Publication Ann. Conf. Genetic Evol. Comput.*, 2007, pp. 2933–2936.

[20] C. Alexander, *A Pattern Language: Towns, Buildings, Construction*. Oxford, U.K.: Oxford University Press, 1977.

[21] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, *Design Patterns: Elements of Reusable Object-Oriented Software*. Boston, MA, USA: Addison-Wesley Pearson, 1995.

[22] M. Harman, “The role of artificial intelligence in software engineering,” in *Proc. 1st Int. Workshop Realizing AI Synergies Softw. Eng.*, 2012, pp. 1–6.

[23] B. A. Kitchenham, D. Budgen, and P. Brereton, *Evidence-Based Software Engineering and Systematic Reviews*. Boca Raton, FL, USA: CRC Press, 2016.

[24] A. M. Pitangueira, R. S. P. Maciel, and M. Barros, “Software requirements selection and prioritization using SBSE approaches: A systematic review and mapping of the literature,” *J. Syst. Softw.*, vol. 103, pp. 267–280, 2015.

[25] O. Räihä, “A survey on search-based software design,” *Comput. Sci. Rev.*, vol. 4, no. 4, pp. 203–249, 2010.

[26] A. Aleti, B. Buhnova, L. Grunske, A. Koziolek, and I. Meedeniya, “Software architecture optimization methods: A systematic literature review,” *IEEE Trans. Softw. Eng.*, vol. 39, no. 5, pp. 658–683, May 2013.

[27] W. Afzal, R. Torkar, and R. Feldt, “A systematic review of search-based testing for non-functional system properties,” *Inf. Softw. Technol.*, vol. 51, no. 6, pp. 957–976, 2009.

[28] G. Bavota, M. Di Penta, and R. Oliveto, *Search Based Software Maintenance: Methods and Tools*. Berlin, Heidelberg, Germany: Springer, 2014, pp. 103–137.

[29] T. Mariani and S. R. Vergilio, “A systematic review on search-based refactoring,” *Inf. Softw. Technol.*, vol. 83, pp. 14–34, 2017.

[30] A. S. Sayyad and H. Ammar, “Pareto-optimal search-based software engineering (POSBSE): A literature survey,” in *Proc. 2nd Int. Workshop Realizing Artif. Intell. Synergies Softw. Eng.*, 2013, pp. 21–27.

[31] M. Harman, “SBSE: Introduction, Motivation, Results and Directions (keynote),” in *Proc. 6th Int. Symp. Search Based Softw. Eng.*, Fortaleza, Brazil, Aug. 2014, doi: 10.1007/978-3-319-09940-8.

[32] M. Harman, Y. Jia, and Y. Zhang, “Achievements, open problems and challenges for search based software testing,” in *Proc. 8th IEEE Int. Conf. Softw. Testing Verification Validation*, 2015, pp. 1–12.

[33] C. Simons, J. Singer, and D. R. White, “Search-based refactoring: Metrics are not enough,” in *Proc. 7th Int. Symp. Search-Based Softw. Eng.*, 2015, pp. 47–61.

[34] G. R. Santhanam, “Qualitative optimization in software engineering: A short survey,” *J. Syst. Softw.*, vol. 111, pp. 149–156, 2016.

[35] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*. Upper Saddle River, NJ, USA: Prentice-Hall, 2009, pp. 64–202.

[36] C. Blum and A. Roli, “Metaheuristics in combinatorial optimization: Overview and conceptual comparison,” *ACM Comput. Surv.*, vol. 35, no. 3, pp. 268–308, 2003.

[37] I. Boussaïd, J. Lepagnot, and P. Siarry, “A survey on optimization metaheuristics,” *Inf. Sci.*, vol. 237, pp. 82–117, 2013.

[38] B. Li, J. Li, K. Tang, and X. Yao, “Many-objective evolutionary algorithms: A survey,” *ACM Comput. Surv.*, vol. 48, no. 1, pp. 13:1–35, 2015.

[39] A. Zhou, B.-Y. Qu, H. Li, S.-Z. Zhao, P. N. Suganthan, and Q. Zhang, “Multiobjective evolutionary algorithms: A survey of the state of the art,” *Swarm Evol. Comput.*, vol. 1, no. 1, pp. 32–49, 2011.

[40] I. C. Parmee, “Poor-definition, uncertainty, and human factors-satisfying multiple objectives in real-world decision-making environments,” in *Proc. Int. Conf. Evol. Multi-Criterion Optim.*, 2001, pp. 52–66.

[41] M. Ware, E. Frank, G. Holmes, M. Hall, and I. H. Witten, “Interactive machine learning: Letting users build classifiers,” *Int. J. Human.-Comput. Stud.*, vol. 55, no. 3, pp. 281–292, 2001.

[42] B. Kitchenham and S. Charters, “Guidelines for performing systematic literature reviews in software engineering,” Keele University and Durham University, Durham, U.K., Tech. Rep. EBSE-2007-01, 2007.

[43] N. Salleh, E. Mendes, and J. Grundy, “Empirical studies of pair programming for CS/SE teaching in higher education: A systematic literature review,” *IEEE Trans. Softw. Eng.*, vol. 37, no. 4, pp. 509–525, Jul./Aug. 2011.

[44] A. M. Pitangueira, R. S. P. Maciel, M. de Oliveira Barros, and A. S. Andrade, “A systematic review of software requirements selection and prioritization using SBSE approaches,” in *Proc. 5th Int. Symp. Search Based Softw. Eng.*, 2013, pp. 188–208.

[45] R. S. Pressman and B. R. Maxim, *Software Engineering: A Practitioner’s Approach*. New York, NY, USA: McGraw-Hill Education, 2014.

[46] D. Zhang and J. J. Tsai, “Machine learning and software engineering,” *Softw. Quality J.*, vol. 11, no. 2, pp. 87–119, 2003.

[47] A. Arcuri and L. Briand, “A hitchhiker’s guide to statistical tests for assessing randomized algorithms in software engineering,” *Softw. Test. Verif. Rel.*, vol. 24, no. 3, pp. 219–250, 2014.

[48] A. A. Araújo and M. Paixão, “Machine learning for user modeling in an interactive genetic algorithm for the next release problem,” in *Proc. 6th Int. Symp. Search-Based Softw. Eng.*, 2014, pp. 228–233.

[49] B. Marculescu, R. Feldt, and R. Torkar, “Practitioner-oriented visualization in an interactive search-based software test creation tool,” in *Proc. 20th Asia-Pacific Softw. Eng. Conf.*, 2013, vol. 2, pp. 87–92.

[50] B. Marculescu, R. Feldt, and R. Torkar, “A concept for an interactive search-based software testing system,” in *Proc. 4th Int. Symp. Search Based Softw. Eng.*, 2012, pp. 273–278.

[51] P. Tonella, A. Susi, and F. Palma, “Using interactive GA for requirements prioritization,” in *Proc. 2nd Int. Symp. Search Based Softw. Eng.*, 2010, pp. 57–66.

[52] L. Troiano, C. Birtolo, and G. Cirillo, “Interactive genetic algorithm for choosing suitable colors in user interface,” in *Proc. Int. Conf. Learn. Intell. Optim.*, 2009, pp. 14–18.

[53] C. L. Simons and I. C. Parmee, “Agent-based support for interactive search in conceptual software engineering design,” in *Proc. Ann. Conf. Genetic Evol. Comput.*, 2008, pp. 1785–1786.

[54] C. Simons and I. Parmee, “User-centered, evolutionary search in conceptual software design,” in *Proc. IEEE Congr. Evol. Comput.*, 2008, pp. 869–876.

[55] N. Monmarché, G. Nocent, G. Venturini, and P. Santini, "On generating HTML style sheets with an interactive genetic algorithm based on gene frequencies," in *Proc. 4th Eur. Conf. Artif. Evol.*, 1999, pp. 99–110.

[56] M. Jackson, "Automated software engineering: Supporting understanding," *Automat. Softw. Eng.*, vol. 15, no. 3, pp. 275–281, 2008.

[57] S. G. Hart and L. E. Staveland, "Development of NASA-TLX (Task Load Index): Results of empirical and theoretical research," *Adv. Psychol.*, vol. 52, pp. 139–183, 1988.

[58] A. Bryant and K. Charmaz, *The Sage Handbook of Grounded Theory*. Newbury Park, CA, USA: Sage, 2007.

[59] M. Myers, "Qualitative research in information systems," *Manag. Inf. Syst. Quarterly*, vol. 21, no. 2, pp. 241–242, 1997.

[60] A. Strauss and J. Corbin, *Basics of Qualitative Research*. Newbury Park, CA, USA: Sage, 1990.

[61] A. Kosorukoff, "Human based genetic algorithm," in *Proc. IEEE Int. Conf. Syst. Man Cybern*, 2001, vol. 5, pp. 3464–3469.

[62] B. J. Bush and H. Sayama, "Hyperinteractive Evolutionary Computation," *IEEE Trans. Evol. Comput.*, vol. 15, no. 3, pp. 424–433, Jun. 2011.

[63] J. Byrne, E. Hemberg, M. ONeill, and A. Brabazon, "A methodology for user directed search in evolutionary design," *Genet. Program. Evol. Mach.*, vol. 14, no. 3, pp. 287–314, 2013.

[64] M. Shackelford and C. L. Simons, "Metaheuristic design pattern: Interactive solution presentation," in *Proc. Companion Publication Ann. Conf. Genetic Evol. Comput.*, 2014, pp. 1431–1434.

[65] A. Ramírez, J. R. Romero, and S. Ventura, "A comparative study of many-objective evolutionary algorithms for the discovery of software architectures," *Empir. Softw. Eng.*, vol. 21, no. 6, pp. 2546–2600, 2016.

[66] R. M. Hierons, M. Li, X. Liu, S. Segura, and W. Zheng, "SIP: Optimal product selection from feature models using many-objective evolutionary optimization," *ACM Trans. Softw. Eng. Methodol.*, vol. 25, no. 2, pp. 17:1–17:39, 2016.

[67] M. W. Mkaouer, M. Kessentini, S. Bechikh, M. Ó Cinnéide, and K. Deb, "On the use of many quality attributes for software refactoring: A many-objective search-based software engineering approach," *Empir. Softw. Eng.*, vol. 21, no. 6, pp. 2503–2545, 2016.

[68] A. P. Piotrowski, M. J. Napiorkowski, J. J. Napiorkowski, and P. M. Rowinski, "Swarm intelligence and evolutionary algorithms: Performance versus speed," *Inf. Sci.*, vol. 384, pp. 34–85, 2017.

[69] C. Simons and J. Smith, "A comparison of meta-heuristic search for interactive software design," *Softw. Comput.*, vol. 17, no. 11, pp. 2147–2162, 2013.

[70] J. J. Durillo and A. J. Nebro, "jMetal: A Java framework for multi-objective optimization," *Adv. Eng. Softw.*, vol. 42, no. 10, pp. 760–771, 2011.

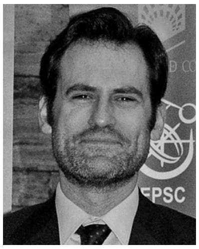
[71] A. Ramírez, J. R. Romero, and S. Ventura, "An extensible jclec-based solution for the implementation of multi-objective evolutionary algorithms," in *Proc. Companion Publication Ann. Conf. Genetic Evol. Comput.*, 2015, pp. 1085–1092.

[72] G. Neumann, J. Swan, M. Harman, and J. A. Clark, "The executable experimental template pattern for the systematic comparison of metaheuristics," in *Proc. Companion Publication Ann. Conf. Genetic Evol. Comput.*, 2014, pp. 1427–1430.

[73] A. Aleti and I. Meedeniya, "Component deployment optimisation with Bayesian learning," in *Proc. 14th Int. Symp. Component-Based Softw. Eng.*, 2011, pp. 11–20.

[74] M. Shackelford and D. Corne, "Collaborative evolutionary multi-project resource scheduling," in *Proc. IEEE Congress Evol. Comput.*, 2001, vol. 2, pp. 1131–1138.

Aurora Ramírez received the MSc degree in computer science from the University of Córdoba, Spain, in 2012. Since 2012, she has been with the Knowledge Discovery and Intelligent Systems Research Laboratory of the University of Córdoba, where she is currently working towards obtaining the PhD degree. Her research interests include search-based software engineering, metaheuristics, and data mining.



José Raúl Romero received the PhD degree from the University of Málaga, in 2007. He was an IT consultant. He is associate professor with the University of Córdoba, Spain. His current research interests include the the development of intelligent systems for industrial use, search-based software engineering and data science. He is a member of the IEEE, ACM, and the Spanish Technical Normalization Committee AEN/CTN 71/SC7 of AENOR. He can also be reached at http://www.jrromero.net.

Christopher L. Simons received the PhD degree in interactive evolutionary computation, from the University of the West of England (UWE), Bristol, United Kingdom. He is senior lecturer with the University of the West of England (UWE), Bristol, United Kingdom. Prior to joining University of the West of England in 2002, his practiced as a software engineer, architect and consultant. He is a member of the British Computer Society. His research interests center on interaction and mutual learning between software engineers and interactive computational intelligence.