A Systematic Review of Interaction in Search-Based Software Engineering
— *Classification guide* —

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

November 5, 2017

Abstract

This technical report is provided as additional material to be read together with "A Systematic Review of Interaction in Search-Based Software Engineering". Classifying the data and features extracted from the primary studies is a key part of the systematic review process. Such an analysis is performed in the light of the research questions posed and the broader research literature. For the sake of brevity, this technical report describes and explains the classification schema applied to the extracted data and features, especially adapted for interactive search-based software engineering.

1 Introduction

In the paper "A systematic Review of Interaction in Search-Based Software Engineering" (SBSE), the role of the human software engineer interaction within search-based approaches is reviewed. In the Introduction section of the paper, the motivation for the review is explained as a need to better understand the ways in which interactive SBSE approaches are being researched and have been applied in the field. To steer the direction of the review, two research questions are formulated:

- **RQ1:** In what ways has interactivity been adopted within search-based software engineering?
- **RQ2:** Which findings about search techniques and interactive approaches along the complete development cycle can be extracted from the current state-of-the-art?
- **RQ3:** To what extent do the detected gaps hamper human interaction in SBSE?
- **RQ4:** What are the emerging trends and how might they be addressed in the future?

Drawing upon evidence-based software engineering and systematic review sources [1, 2], several areas of study data have been identified for extraction and analysis. The areas

*Dept. of Computer Science and Numerical Analysis, University of Córdoba, Córdoba, Spain. Email: {aramirez, jrromero}@uco.es

†Dept. of Computer Science and Creative Technologies, University of the West of England, Bristol, United Kingdom. Email: chris.simons@uwe.ac.uk

Table 1: Areas of interest to classify iSBSE studies

Area	**Classifier**
Problem formulation	Type of software engineering problem
	Development practice
	Number of objectives
	Constraints
Search technique	Algorithm type
	Objective approach
Interactive approach	Interactive algorithm type
	User feedback type
	Evaluation mechanism
	User's intervention
	Influence of user's opinion
Experimental framework	Type of study
	User's profile
	Evaluation criteria
	Case studies
	Evidence
	Statistical tests

related to the contextual data of the primary sources, e.g. publication details and problem formulation, together with qualitative data, e.g. the types of search algorithm, interactive approach, etc., and quantitative data, e.g. the number of objectives in search, the number of participants in empirical studies, etc. Thus, the primary sources are analysed with respect to four areas of information gathering, namely:

- Problem formulation;
- Search technique proposed;
- Interactive approach employed; and
- Experimental framework.

For each area of information gathering, the review-specific data to be extracted and classified are specified as follows:

- Goal – the purpose of the classification
- Sources – references relevant to this classification
- Baseline classification – drawing upon the start-of-the-art references above
- Classification – specified classification for review-specific data

The four areas of information gathering, together with their associated classifiers are shown in Table 1.

2 How to use this guide

The aim of this report is to provide an accurate classification scheme to support the extraction process, as well as improve the readability of the analysis results. Unless explicitly indicated, primary studies should fall into one item for every category. When required, the following values will be accepted for any category:

- Not specified: the primary study does not provide information with respect to the specific category.
- Not applicable: the classification cannot be applied to the primary study for a specific reason (to be specified in the comments).

In addition, this guide is intended to help other researchers interested in iSBSE to precisely define the key elements of their proposals. In that case, the following reference can be used to cite this classification guide:

> A. Ramírez, J.R. Romero, C.L. Simons. "Guide for the classification of interactive search-based studies". 2017. Available at: `http://www.uco.es/grupos/kdis/sbse/isbse`

3 Information gathering on the problem formulation

3.1 Type of software engineering problem

Goal. To frame the optimisation problem within the software development lifecycle / SBSE subfield.

References. SBSE survey by Harman *et al.* [3], common phases within software engineering development methodologies.

Baseline classification. From [3], which divides SBSE into the following fields: requirements/specifications; design tools and techniques; software/project verification and model checking; distribution, maintenance and enhancement, including modularisation and refactoring; and management, including project planning and cost estimation).

Classification. The following is derived:

- Requirements
- Analysis and design
- Code implementation
- Testing and verification
- Distribution and maintenance
- Project management

3.2 Development practice

Goal. To contextualise the optimisation approach into a software engineering methodology.

References. Reference book in Software Engineering by Pressman and Maxim [4].

Baseline classification. Pressman and Maxim propose the following classification: prescriptive process models (waterfall, incremental process, evolutionary process–including prototyping and spiral–and concurrent); specialised process models (component-based development, formal methods, aspect-oriented software development); the unified process; agile development (scrum, extreme programming, etc.)

Classification. The following is adopted:

- Prescriptive process model: waterfall, incremental, etc.
- Specialised process model: component-based software engineering, model-driven engineering, etc.
- Agile development: scrum, extreme programming, etc.

3.3 Number of objectives

Goal. To classify the problem according to the number of objectives defined.

References. Survey by Lin *et al.* [5].

Classification. The following is proposed (selecting multiple values is permitted):

- Single-objective: 1 objective.
- Multi-objective: from 2 to 3 objectives.
- Many-objective: greater than 3 objectives.

3.4 Constraints

Goal. To classify the problem according to the presence of constraints.

Classification. The following is proposed:

- Unconstrained problem.
- Constrained problem.

4 Information gathering on the search technique

4.1 Algorithm type

Goal. To classify the resolution technique applied to the optimisation problem.

References. Surveys on search techniques [6], metaheuristics [7] and machine learning [8].

Baseline classification. Search techniques can be classified as follows [6]:

- Exact: "a procedure that guarantees finding an optimal solution."
- Heuristic: "an approximate optimisation procedure, usually dependent on the optimisation problem, which generates a solution by applying a limited set of rules."
- Metaheuristic: "a problem-independent heuristic search strategy that is based on the exploration of the solution space."

Metaheuristics are usually classified according to the number of solutions the method manages [7], thus distinguishing between single-solution based metaheuristics and population-based metaheuristics. Both categories are can be further divided considering the specific search strategy, e.g. simulated annealing, evolutionary computation or swarm intelligence.

Zhang and Tsai (2003) provide a summary of machine learning techniques applied to Software Engineering. Although a specific classification of machine learning techniques is out of scope of the review, some primary studies use machine learning mechanisms. These techniques are usually classified as supervised or unsupervised methods, depending on whether class labelling information is available for training or not.

Classification. The following is derived:

- Exact procedure
- Heuristic procedure
- Metaheuristic procedure
  - Single-solution metaheuristic
  - Population-based metaheuristic
    - Evolutionary computation
    - Swarm intelligence
- Machine leaning
  - Supervised
  - Unsupervised

4.2 Objective approach

Goal. To classify how the search technique treats the problem during its resolution with respect to the number of objectives.

Classification. The following is proposed:

- Single-objective: 1 objective (including the aggregation of n objectives).
- Multi-objective: 2-3 objectives that are optimised independently.
- Many-objective: more than 3 objectives that are optimised independently.

5 Information gathering on the interactive approach

5.1 Interactive algorithm type

Goal. To categorise the algorithm from the interactivity point of view.

References. Taxonomy proposed by Meignan *et al.* [6].

Baseline classification. Meignan *et al.* describes the following approaches:

- Trial and error. "A trial-and-error method can be defined as an iterative and direct adjustment of the optimisation system by the user. More precisely, the user feedback is not generalized and corresponds to the preference information used by the optimisation system. In addition, each iteration of the optimisation process is independent of the previous one. A common example is when the user modifies optimisation parameters and restarts the optimisation procedure."
- Interactive reoptimisation. "(...) interactive reoptimisation aims only at refining the optimisation problem. In interactive reoptimisation, the refinement is done by progressively adjusting a candidate solution using a reoptimisation procedure. (...) Starting from an initial candidate solution, the interactive reoptimisation process alternates between two phases. First, the user specifies changes to be made on the current solution. Second, a reoptimisation procedure is applied to perform the changes and to optimise the rest of the solution accordingly. (...) For the reoptimisation, some specific objectives or constraints are added to the problem model.".
- Interactive multiobjective optimisation. "The preferences provided by the user during the decision-making process should allow the determination of an adequate solution within this set. (...) In the first step, the user evaluates some proposed candidate solutions or provides other preference information with regard to the proposed solutions. This information is used to generate or update the preference model. In the second step, new candidate solutions are generated based on the updated preference model and then proposed to the user. (...) Besides, in interactive approaches, the user may guide the search toward interesting regions of the Pareto-front."
- Interactive evolutionary algorithms. "Within the evolutionary loop (i.e., recombination, mutation, and selection), the user interacts with the system to provide the evaluation of the solutions according to one's own perception of the quality of the solutions. The evaluation provided by the user at each generation is used to select the solutions at the origin of the next generation. (...) The interaction between the user and the system allows the introduction of subjective optimisation criteria that could not be formulated explicitly or identified before the decision-making process. The subjective evaluation of solutions can rely, for instance, on the aesthetic judgement of the user."
- Human-guided search. "In the human-guided search approach, the user provides information that guides the optimisation process (...) and that has no direct impact on the optimisation model. (...) optimisation methods considered in human-guided search are local-search procedures such as hill-climbing

(...) and tabu-search (...). The interactive process alternates between the application of this local search procedure and a feedback session in which the user can express some preferences on how to improve the current solution".

- Additional interactive optimisation approaches[1]: asynchronous teams, crowdsolving (crowdsourcing for solving complex optimisation problems), interactive parameter tuning, hyperinteractive evolutionary computation, long-term preference inference.

Classification. The following definitions are independent of both the search technique and the number of objectives. In addition, a primary study might fall into multiple categories.

- Interactive reoptimisation. Similar to the category with the same name in [6], it is aimed at interactively refining the definition of the optimisation problem. The feedback is used to modify some elements of the problem (constraints, weights, new objectives) instead of the solution itself.
- Preference-based interactivity. Preference information refers here to those decisions that the human makes in light of the intermediate results to guide the search towards some zones of the search/objective space. It includes the elicitation of reference points in multi-objective optimisation, the comparison of solutions to choose the best alternative, the selection of solutions to create an external archive, and similar actions.
- Human-based evaluation. Based on the former category named "interactive evolutionary algorithms" [6], its goal is to (totally or partially) replace the fitness function by the human subjective judgement. Different evaluation mechanisms (metrics, scores, rankings...) might be considered.
- Human-guided search. As defined by Meignan *et al.* [6] but without assuming that a local search technique should be applied. In general, this approach is focused on actions performed by the human that have a direct impact on the solution shown, and probably on other solutions. Examples of possible transformations of the solutions are freezing parts of the encoding or modifying parts of it.

If required, those approaches that do not fall into any of the previous categories would be categorized as *Others*, and a short description should be included.

5.2 User feedback type

Goal. To indicate the kind of information that the user provides to the optimisation algorithm.

References. Review by Meignan *et al.* [6].

Baseline classification. This category is similar to Section 5.1 ("purpose of the interaction and role of the user") in [6]. They divide the interactivity between *problem-oriented* and *search-oriented*:

[1] Definitions have been omitted since these categories come from too specific examples that do not appear in our primary studies.

- Roles of the user in a problem-oriented interaction. "A problem-oriented interaction primarily aims at modifying the optimisation model in order to better fit the decision maker's problem. For this type of interaction, the user's preferences complete the definition of the optimisation problem."
  - Adjuster. "The user is an adjuster when adjusting constraints or objectives. In this case, the type and extent of the modifications made by a user are defined during the design of the optimisation system."
  - Enricher. "The user plays this role when modifying the initial definition of the optimisation problem by adding or removing some constraints or objectives. In this case, it is assumed that the initial problem model may be incomplete, and a candidate solution may not be a valid alternative."
- Roles of the user in a search-oriented interaction: "A search-oriented interaction aims at improving the efficiency of the optimisation procedure. (...). The feedback provided by the user impacts the optimisation procedure without changing the constraints and objectives of the optimisation model."
  - Assistant. "The assistant role is played when the user assists the search process by acting as a search procedure that generates, selects, or modifies solutions."
  - Guide. "The guide role is characterised by the fact that the user modifies the behaviour of the search procedures and controls the exploration of the search space."
  - Tuner. "In this case, the user has no direct control on the operations of the optimisation procedure, but the interaction allows the user to adjust some strategic parameters of the search procedure."

Classification. For the *purpose of user interaction* and the *role of the user*, the above classification is adopted. For the former, both options might be selected if the primary study is categorised into more than one interactive approach. For the latter, more than one option can be selected if the user is allowed to perform more than one action. In addition, the *type of user task* can be classified as follows:

- Evaluation of solutions. The user is asked to evaluate some aspects of the solution quality.
- Selection of solutions. The user selects solutions to locate promising parts of the search space or to indicate that they present specific characteristics of his/her interest.
- Comparison of solutions. The user is asked to compare two or more solutions.
- Modification of solutions. The user manually corrects the solution or freezes some parts of it.

Similarly, multiple values can be selected here.

5.3 Evaluation mechanism

Goal. To characterise the nature of the evaluation provided by the human. Therefore, this category should be only considered whenever the user has a direct (or indirect)

influence in the evaluation phase, without assuming a specific interactive approach (see Section 5.1).

Classification. The following is proposed:

- Fitness value: the user provides a value for the metric used to define the fitness function.
- Weights: the user assigns weights to the objective functions of a multi/many-objective problem.
- Scores: the user rates the solution according to a discrete range of values.
- Rankings: the user sorts a set of solutions according to their quality.
- Reward/penalisation: the user's opinion is used to promote or penalise the quality of a solution.

5.4 User's intervention

About the solutions shown

Goal. To describe the mechanism used to select the solutions. It includes the number of solutions shown, the part of the solutions shown and the responsible for the selection.

Classification. The following is proposed:

- *Number of solutions*:
  - One solution
  - Pair of solutions (probably to be compared)
  - N solutions (fixed number or percentage)
  - All solutions
- *Level of detail*:
  - Complete solution
  - Partial solution
- *Selection strategy*:
  - Fixed: the algorithm chooses the solution(s). It might be refined into:
    - Best solution(s)
    - Random solution(s)
    - Specific solution(s): a different, and probably more specific, selection criterion than the previous ones (clustering, combined criteria...)
    - All solutions
  - Free: the user can freely choose the solution(s).

Adjustment of interaction time

Goal. To describe the mechanism used to decide the specific moment in which each interaction happens.

Classification. The following is proposed:

- Fixed: the iterative scheme is set a priori.
  - Every iteration
  - Every N iterations
  - Between two runs (the same of different algorithms)
- Dynamic: the interactive steps are decided on runtime.
  - Adaptive: the algorithm decides when the user should interact based on the search process.
  - On demand: the user stops the process to provide feedback at certain moments of the search.

5.5 Influence of user's opinion

Goal. To describe how the information is integrated in the search process and its influence in the subsequent search iterations.

References. Review by Meignan *et al.* [6].

Baseline classification. According to Meignan *et al.* [6], two different aspects can be considered within this category:

- *Feedback integration.* "The integration of the user's feedback into the preference model. This integration can be done in two different ways. The user's feedback can be either directly used to compute preference information or the feedback can be generalized through the preference model."
  - Model-free. "In a model-free approach, the values of the preference model are directly updated with the feedback of the user. (...) For instance, in most interactive methods for multiobjective optimisation, the values provided by the user are directly used in the objective function."
  - Model-based. "In a model-based approach, a model of the feedback (or model of the user's preferences) is learned; this model is used to derive the preference information. (...) A major interest of a model-based approach is the possibility to derive potentially complex preference information from simple feedback from the user."
- *Preference information lifetime.* "It is the period of validity of the information contained in the preference model."
  - Step-based. "Step-based preference information is specific to a part of the optimisation process. The information may not be valid for the whole optimisation process and it must be redefined at different stages."
  - Short-term. "Short-term preference information is potentially valid for the whole optimisation process, but cannot be reused for solving another problem instance."
  - Long-term. "Long-term preference information can possibly be reused for solving different problem instances."

Classification. The above categories are adopted in order to characterise the influence of user's opinion. Additionally, the *information validity* is classified as follows:

- Permanent: once the feedback provided by the human is integrated, it lives on the entire optimisation process without any change.
- Flexible: the human can modify his/her decision in the following interactions, e.g. weight reconfiguration.
- Unrestricted: the human can revoke his/her decisions during the interaction, e.g. unfreezing a part of the solution that was previously frozen, removing a constraint added by him/her.

As different kind of information might be gathered from the user at the same time, more than one option could be selected here if they were treated independently.

6 Information gathering on the experimental framework

6.1 Type of study

Goal. To identify the scope of the experiments carried out by each primary study.

Classification. The following categories are proposed:

- Theoretical proposal: the paper does not include any experimentation.
- Sample execution: an example showing how the algorithm behaves, normally conducted by the authors themselves.
- Simulated interaction: user's interaction is simulated.
- Empirical investigation: there is an empirical study with participants.

The categories are sorted from less realistic/valuable to more realistic/valuable (from the point of view of the interactivity). If multiple experiment appears in a primary study, they all should be reflected in the extraction form.

6.2 User's profile

Goal. To characterise the participants within interactive experiments.

Classification. The following categories are proposed regarding the *number of participants*:

- 1 participant
- Between 2 and 10 participants
- Between 11 and 20 participants
- More than 20 participants

Then, the presence of *user's profile* information should be indicated as follows:

- Yes
- No
- Partially

If the latest response is "yes" or "partially", the following fields should be specified:

- Position:
  - Engineer (industry)
  - PhD academia staff (lecturer, researcher, etc.)
  - Postgraduate students (PhD students or master students)
  - Undergraduate students
  - Other (specify in comments)
- Expertise (in software development/engineering)
  - Less than 5 years
  - Between 5 and 10 years
  - Between 11 and 20 years
  - More than 20 years

For the position and expertise fields[2], more than one category can be selected if more than one person participates in the empirical investigation.

6.3 Evaluation criteria

Goal. To classify the mechanism used to analyse the experimental outcomes.

Classification. The following categories are proposed:

- Measures: fitness values or specific assessment metrics are reported.
- Solutions: visualisation of solutions generated during the interactive session, or comparisons to manually/automatically produced solutions in terms of metrics or qualitative aspects.
- Questionnaires: responses to questions or general comments about the interactive session are detailed.

Multiple options can be selected for every category.

6.4 Case studies

Goal. To classify the magnitude and scope of the software artefacts used as inputs.

Classification. The following categories are proposed:

- Synthetic: the problem instance was artificially generated.
- Controlled environment: the problem instance represents a relatively small, but real, problem that is known by the participants.
- Industrial case: the problem comes from an industry partner.

Multiple options can be selected for every category.

[2]The specific ranges have been defined in the light of the information extracted from the primary studies.

6.5 Evidence

Goal. To indicate whether additional materials, especially those related to the interactive sessions (questionnaires, transcripts, solutions), are publicly available.

Classification. Firstly, the availability of additional material should be indicated as:

- Yes
- Yes, but unreachable
- No

If the response is "yes", the web page should be indicated as a comment and any available information should the classified according to the following categories (multiple options are possible):

- Source code
- Problem instances / case studies
- Experimental results in raw format (log files, tables, reports, etc.)
- Solutions generated in an interactive session
- Questionnaires (forms not necessarily including the answers)
- Transcripts of the interactive session (audio, video, writing)
- Additional statistics
- Other (to be specified)

6.6 Statistical tests

Goal. To classify the statistical tests used to validate the experimental results, if any.

References. Arcuri and Briand's work on the use of statistical tests in SBSE [9].

Baseline classification. On the one hand, parametric and non-parametric tests are distinguished depending on whether they assume the statistical distribution of the data or not, respectively. On the other hand, statistical tests are usually classified according to the number of algorithms whose performance is under comparison, i.e. pairwise (2) and multiple (>2) comparison. In addition, the use of effect size measurements is recommended in order to assess the magnitude of the improvements.

Classification. The following categories are proposed:

- Pairwise comparison (parametric and non-parametric tests)
- Multiple comparison (parametric and non-parametric tests)
- Effect size measurement (parametric and non-parametric tests)

References

[1] B. Kitchenham and S. Charters, "Guidelines for performing Systematic Literature Reviews in Software Engineering," tech. rep., Keele University and Durham University, 2007.

[2] B. A. Kitchenham, D. Budgen, and P. Brereton, *Evidence-Based Software Engineering and Systematic Reviews*. CRC Press, 2016.

[3] M. Harman, S. A. Mansouri, and Y. Zhang, "Search-based software engineering: Trends, techniques and applications," *ACM Computing Surveys*, vol. 45, no. 1, p. 11, 2012.

[4] R. S. Pressman and B. R. Maxim, *Software Engineering: A Practitioner's Approach*. McGraw-Hill Education, 8th ed., 2014.

[5] B. Li, J. Li, K. Tang, and X. Yao, "Many-Objective Evolutionary Algorithms: A Survey," *ACM Computing Surveys*, vol. 48, no. 1, pp. 13:1–35, 2015.

[6] D. Meignan, S. Knust, J.-M. Frayret, G. Pesant, and N. Gaud, "A review and taxonomy of interactive optimization methods in operations research," *ACM Transactions on Interactactive Intelligent Systems*, vol. 5, no. 3, p. 17, 2015.

[7] I. Boussaïd, J. Lepagnot, and P. Siarry, "A survey on optimization metaheuristics," *Information Sciences*, vol. 237, pp. 82–117, 2013.

[8] D. Zhang and J. J. Tsai, "Machine Learning and Software Engineering," *Software Quality Journal*, vol. 11, no. 2, pp. 87–119, 2003.

[9] A. Arcuri and L. Briand, "A Hitchhiker's guide to statistical tests for assessing randomized algorithms in software engineering," *Software Testing, Verification and Reliability*, vol. 24, no. 3, pp. 219–250, 2014.