ISearchAI: A Hybrid Framework for Interactive Search-Based Software Engineering with Machine Learning

Willian M. Freire
Federal University of Technology-Paraná
Campo Mourão, Brazil
willianfreire@utfpr.edu.br

Aline M. M. M. Amaral
State University of Maringá
Maringá, Brazil
ammmamaral@uem.br

Thelma E. Colanzi
State University of Maringá
Maringá, Brazil
thelma@din.uem.br

Abstract

Search-Based Software Engineering (SBSE) has proven effective in solving complex search problems, yet purely automated results often diverge from human expectations due to the difficulty of formalizing subjective preferences. Interactive SBSE (iSBSE) addresses this by incorporating the Decision Maker (DM) into the search process, but its development is currently hindered by technical barriers and the risk of human fatigue during the evaluation of optimized solutions. This paper presents *ISearchAI*, a hybrid research framework designed to facilitate the development of iSBSE approaches that integrate Machine Learning (ML). The proposed framework employs a decoupled, three-layer architecture that separates frontend interaction from backend search and from external ML and search libraries, thereby promoting scientific reproducibility and architectural extensibility. To evaluate the framework, two functional case studies were conducted: the automated generation of an iSBSE approach for the Next Release Problem (NRP) and the adaptation of a consolidated search tool for Software Product Line Architectures (PLA). Quantitative results indicate that *ISearchAI* supports the execution of interactive workflows in a controlled timeframe and achieves predictive accuracy exceeding 96% with ML models. Furthermore, statistical analysis demonstrates that the framework's abstraction layer is non-intrusive, preserving the integrity and performance of consolidated tools. This work establishes a standardized research infrastructure for iSBSE, enabling investigators to focus on preference modeling rather than on the engineering overhead of research infrastructure.

Keywords

Interactive Search-Based Software Engineering, Machine Learning, Framework

1 Introduction

Search-Based Software Engineering (SBSE) has established itself as a means of solving complex search problems throughout the software development lifecycle by applying metaheuristic algorithms to high-dimensional search spaces [22]. By transforming software engineering tasks into search problems, SBSE aims to identify optimal or near-optimal solutions for challenges such as software design, testing, and maintenance [8]. However, despite the computational efficiency of automated search, a recurrent challenge persists: the solutions identified by algorithms often diverge from the practical expectations of software engineers due to the inherent difficulty of formalizing qualitative and subjective nuances into mathematical objective functions [26].

To address this limitation, Interactive Search-Based Software Engineering (iSBSE) incorporates the Decision Maker (DM) directly into the search process to capture subjective preferences in real-time [4]. This approach enables the DM to guide the search by interactively evaluating optimized solutions [20]. Hence, solutions generated through interactive search tend to achieve more acceptance and practical applicability than those produced by automated, non-interactive metaheuristics that lack human insight [27].

Despite its benefits, the development of iSBSE approaches remains an engineering task that imposes technical overhead and implementation costs on researchers [17]. Software engineers frequently face a barrier to entry due to the lack of standardized research infrastructure that facilitates the development of interactive approaches [11]. This fragmentation results in ad hoc implementations that limit replication and systematic comparison across problem domains [12].

A primary constraint in interactive approaches is human fatigue, characterized by cognitive stress experienced by the DM when evaluating numerous optimized solutions [30]. Literature indicates that as fatigue increases, the consistency of human feedback tends to decrease, introducing noise and bias into the search process [32]. This phenomenon highlights the need for mediation mechanisms that can capture preferences without overwhelming the human participant (in this case, the DM) during the search [5].

In the context of iSBSE, Machine Learning (ML) algorithms are used to mitigate fatigue by providing an ML-assisted iSBSE paradigm [2]. By learning preference patterns from a set of human evaluations of solutions during the search process, ML models can approximate the DM's evaluation, thereby partially automating interactions and allowing the search to proceed with reduced human intervention while remaining aligned with DM's criteria [1]. But, the native integration of ML into search-based approaches is still undergoing development, often requiring researchers to manually bridge disparate libraries and handle data transformations [1].

Existing frameworks, such as jMetal and Nautilus, provide the computational infrastructure for developing SBSE approaches and tools [9, 10]. Specifically, jMetal offers a comprehensive suite of search algorithms, whereas Nautilus provides mechanisms for user interaction. However, neither framework offers native, end-to-end support for ML-assisted interactive SBSE (iSBSE) nor dual-workflow mechanisms that simultaneously support approach generation and tool adaptation [10].

In this context, this work introduces *ISearchAI*, a hybrid and extensible framework that serves as a standardized research infrastructure for ML-assisted iSBSE. *ISearchAI* decouples the domain logic from library implementations through a layered architecture,

integrating search and ML algorithms with interaction handlers [16]. The framework was designed to facilitate the development of iSBSE approaches and to support methodological consistency through evaluation modules and support for statistical validation.

The primary objective of this work is to present the *ISearchAI* framework and the evaluation of its technical feasibility through two distinct validation workflows: (1) the automated generation of new approaches and (2) the adaptation of an existing tool. Each workflow was evaluated through a specific case study designed to demonstrate the framework's versatility across different problem domains and implementation needs.

To evaluate the framework's capability to support the creation of new approaches from conceptual definitions (Generation Workflow), Case Study 1 applies *ISearchAI* to the Next Release Problem (NRP). NRP is a representative NP-complete combinatorial optimization challenge focused on prioritizing requirements in development cycles. This study evaluated how the framework's support mechanisms facilitate the implementation of a functional iSBSE approach for requirements selection, where customer importance and implementation costs are simultaneously optimized.

To evaluate the feasibility of extending and retrofitting established search-based software tools (Adaptation Workflow), Case Study 2 focused on Software Product Line Architecture (PLA) design. This domain is characterized by architectural complexity and the need to manage points of variation. In this context, the study investigated the framework's capacity to integrate interaction and ML mechanisms into a consolidated tool without compromising its original search logic or requiring a fundamental rewrite of the existing source code. This case study verified the framework's architectural non-intrusiveness when integrated into established infrastructures.

Therefore, this research utilized both case studies to quantitatively evaluate whether the proposed framework reduces technical barriers to developing interactive approaches. Through functional and statistical validation, we demonstrate the framework's feasibility in generating new iSBSE approaches and adapting existing tools. The contributions of this research are: (i) the proposal of a hybrid framework for ML-assisted iSBSE that integrates machine learning with interactive search; (ii) the empirical evaluation of the Generation Workflow, executed through a controlled user experiment with 11 participants in the NRP domain; and (iii) the validation of the Adaptation Workflow, conducted via an engineering case study to evaluate the architectural integration within a consolidated PLA design tool. Collectively, this research establishes a standardized baseline for iSBSE, enabling a focus on preference modeling rather than on the implementation overhead of software integration.

The remainder of this work is organized as follows. Section 2 provides related works. Section 3 details the *ISearchAI* framework. Section 4 describes the experimental design. Section 5 presents the quantitative results. The remaining sections present threats to validity and concluding remarks.

2 Related Work

This section presents the state of the art in libraries and frameworks that support the development of iSBSE and SBSE approaches and tools. The solutions are classified according to their technical characteristics and primary application objectives.

Two of the most widely adopted frameworks are jMetal [9] and the MOEA Framework [21]. jMetal is an object-oriented library that provides a modular architecture for defining optimization problems, search algorithms, and quality indicators. Similarly, the MOEA Framework offers a comprehensive environment for developing and evaluating multi-objective optimization approaches through a broad collection of evolutionary and metaheuristic algorithms.

PISA (Platform and Programming Language Independent Interface for Search Algorithms) [6] is an interface specification, designed to separate the search algorithm from the problem-specific part via a text-based communication protocol, enabling multi language integration. ECJ (Evolutionary Computation in Java) [29] is a toolkit emphasizing genetic programming and general evolutionary computation. PlatEMO [34] and DEAP (Distributed Evolutionary Algorithms in Python) [15] provide a collection of multi-objective and many-objective algorithms, primarily used for benchmarking.

Regarding interactivity, it is worth mentioning DESDEO [25] and Nautilus [10]. DESDEO is an open-source framework for interactive multiobjective optimization, focusing on decision-making methods that enable users to express preferences through objective aspirations. Nautilus expands this concept by providing a plug-and-play Java web platform that integrates with jMetal. It supports DM feedback via visual interfaces and parallelizes evaluations through cloud computing to handle large problem instances. Despite these contributions, the current state of the art lacks a research framework that natively integrates ML models into the search process to mitigate human fatigue and provides automated mechanisms for generating and adapting iSBSE tools.

Compared with these frameworks, *ISearchAI* addresses current gaps in the iSBSE development lifecycle. While platforms like jMetal, MOEA Framework, and PlatEMO provide search algorithms, and Nautilus and DESDEO facilitate human interaction, they do not natively integrate some features, such as ML models, to mitigate human fatigue during the search process. Furthermore, unlike the aforementioned frameworks, which are designed to execute predefined experiments, *ISearchAI* introduces dedicated modules for the automated generation of new iSBSE approaches and the systematic adaptation of existing tools. This framework also incorporates a built-in suite of statistical methods for validating interactive results, a feature not present in existing frameworks.

The distinct engineering scope of *ISearchAI* involves providing a unified representation that natively bridges search spaces and machine learning classifiers. While existing tools focus on standalone search execution (*jMetal*) or independent graphical user evaluation interfaces (*Nautilus*), *ISearchAI* automates code infrastructure generation via concrete package structures and handles legacy code retrofitting via inheritance mechanisms.

3 The *ISearchAI* Framework

This section presents *ISearchAI*, the hybrid framework [1] proposed in this work to assist software engineers in developing iSBSE approaches. *ISearchAI* integrates ML algorithms into the interactive search process, supports multi and many-objective scenarios, and

[1] A hybrid framework in this context refers to an infrastructure that provides ready-to-use components for immediate implementation while remaining extensible to enable customization and the creation of new resources as the research evolves.

provides a standardized layer for search interactions. The following subsections detail the framework´s architecture, graphical interface, and the integrated module for evaluating approach results.

3.1 Architectural Design

The architecture of *ISearchAI* is based on domain decoupling and organized into three distinct layers to promote reproducibility and extensibility (Fig. 1). Reproducibility is facilitated by the strict isolation of experimental variables; by separating the search engine (Layer B) from external libraries (Layer C), researchers can keep the interaction logic constant while swapping only the underlying ML or search algorithms. This prevents the side effects that are common in non-modular implementations, where business logic and infrastructure are intertwined. Extensibility is achieved through the abstraction of solution encodings and the modular integration of external libraries. This allows the framework to handle the increased computational demand of search problems and complex architectures without necessitating changes to the communication protocol or the user interface.

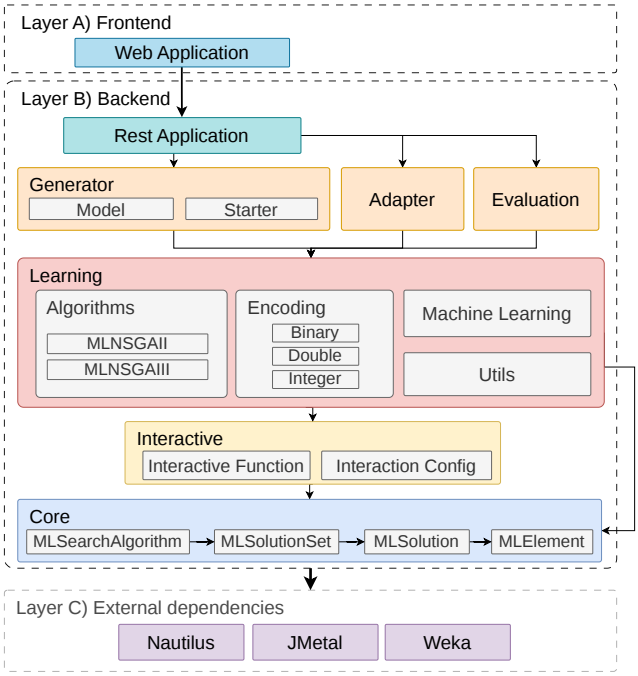

Figure 1: General architecture of the *ISearchAI* framework, illustrating the interaction between modules.

The specific functional responsibilities and technical components of each architectural layer are detailed as follows.

Layer A (Frontend): A ReactJS-based [2] directed interface that guides the Software Engineer through problem configuration. It manages the collection of encoding parameters and interaction strategies, abstracting the implementation details of developing the DM interaction interface for new iSBSE approaches. Subsection 3.3 presents the graphical interface implemented in this layer.

[2]https://react.dev

Layer B (Backend): The core logic layer developed in Java, containing all modules and components used to implement the iSBSE approach. Java was adopted for developing the framework, since many approaches in the literature are implemented in it. The first module, named *Rest Application*, contains the RESTful application that allows the communication between Layers A and B. The *Generator* and *Adapter* modules orchestrate the creation or adaptation of existing iSBSE tools.

The *Evaluation* module provides built-in implementations of metrics and statistical tests commonly used in experimental software engineering, including search metrics, such as automated calculation of Hypervolume (HV) and Euclidean Distance (ED). Also includes metrics for evaluation of ML models via Kappa Coefficient, F1-Score, and accuracy. This module integrates support for normality tests (Shapiro-Wilk) and significance tests (Kruskal-Wallis, ANOVA, and Vargha-Delaney effect size), facilitating the validation of interactive versus non-interactive results [36].

The *Learning* module includes all interactive search algorithms integrated with ML algorithms, available encodings, and utilities that software engineers can use when developing an iSBSE approach. Next, the *Interactive* module contains the specification classes and interfaces that enable the interaction with DMs.

Finally, the *Core* module defines core abstractions, such as *MLSolution* and *MLSearchAlgorithm*, to implement the search solution and its composition, as the selected search algorithm.

To prevent naming conflicts with external libraries sharing common identifiers (e.g., *Solution* in jMetal, Weka, or Nautilus), *ISearchAI* prefixes its core abstractions with *ML* (e.g., *MLSolution*, *MLSearchAlgorithm*). This strategy ensures that the software engineer can clearly distinguish between the framework's core abstractions and implementation-specific classes from third-party dependencies.

Layer C (External Dependencies): This layer integrates established libraries (e.g., JMetal, Weka, and Nautilus) as default implementations. The design utilizes an abstraction layer that treats these libraries as modular components rather than rigid dependencies.

The design focus of *ISearchAI* lies in its multi-tiered abstraction layer rather than the mere orchestration of existing libraries. By treating external dependencies as plug-and-play components, the framework decouples the iSBSE logic from specific tool implementations. This design allows replacement of search or ML algorithms (e.g., swapping Weka for another learning library), since the core interfaces (*MLSearchAlgorithm* and *MLSolution*) are implemented. Consequently, the framework functions as a generative infrastructure that orchestrates non-interactive components into a cohesive interactive search workflow.

To support a diverse range of software engineering problems, the framework employs different solution encodings, which define the programmatic representation of a candidate solution. *ISearchAI* provides native support for multiple encoding types through the *MLSolution* abstraction. This allows the framework to handle both vector-based representations, such as the binary and floating-point encodings used in the NRP, and complex object-oriented encodings required for structural search, as observed in the PLA domain. By decoupling the representation from the search logic, the framework ensures that interaction and machine learning modules remain functional regardless of the underlying problem structure.

3.2 Interactive Search and ML Model Integration

The framework systematizes the interaction between the DM and the search approach via the *Interactive* module in Layer B. The components of this module capture subjective preferences, such as assigning scores [5] or "freezing" [20] specific elements of the solution encoding, and store the resulting data for preference modeling.

The interaction cycle begins when the search process reaches a predefined synchronization point, such as a specific generation. At this stage, the search algorithm is suspended to allow for human intervention. The framework's frontend (Layer A) presents the solutions optimized up to that point. This presentation facilitates a comparative analysis, enabling the DM to identify technical trade-offs (the best cost-benefit) between conflicting objectives before providing subjective feedback.

Under the scoring strategy, the DM performs a global evaluation by assigning discrete numerical values to each solution presented, using a predefined scale to represent perceived quality [5]. These numerical preference scores provide quantitative data that backend components use to rank individuals in the active search population. Furthermore, these scores serve as the primary dataset for the integrated ML-based preference models, allowing the framework to learn the DM's preferences and subsequently automate evaluations in future search iterations to mitigate fatigue.

In contrast, another strategy, called "freezing", enables localized, granular interaction. In this mode, the DM identifies specific solution fragments or attributes, such as a particular requirement selection in NRP or a component placement in PLA, and "freezes" them to ensure they remain unchanged during subsequent search operations [20]. While scoring guides the search by influencing the algorithm, freezing directly constrains the search space, allowing the process to focus on refining the unfrozen variables.

A core feature of *ISearchAI* is its native ML integration to mitigate DM fatigue. Candidate solution encodings (e.g., binary vectors in NRP) map directly into feature attributes for classification. To manage cold-start scenarios, the search initially executes without assistance for N generations to accumulate user preference labels. Automated evaluation triggers after $k = 3$ interactive rounds (or when $M = 60$ labeled instances are collected). The ML default configuration is based on previous works [5, 20].

Automation Mechanisms: *ISearchAI* provides two distinct workflows to support the development of iSBSE approaches:

(1) Generation Module: Facilitates the creation of new approaches through a three-step workflow: (i) *Configuration (Layer A):* The user defines solution encodings, objectives, and interaction mechanisms via the React interface; (ii) *Local Setup (Layer B):* The framework exports a standard Java Maven skeleton [33] containing seven packages (*instance*, *model*, *objective*, *problem*, *runner*, interaction handlers, and configuration mappings), where the developer manually implements the dataset parser in *instance* and objective functions in *objective* within their IDE; (iii) *Execution:* The backend engine executes the search, triggering Layer A synchronization interfaces when human feedback (scoring or freezing) is required. Scoring (macro-level) and freezing (micro-level) serve as baseline constraints, extensible via the *Interactive* module interfaces asd asd asd asd asd asd asd asd asd .

To illustrate the operational steps, consider a running example where an engineer defines a problem template. The module automatically instantiates a seven-package structural skeleton (*instance, model, objective, problem, runner*, interaction handlers, and configuration mappings). The software engineer manually implements only the input dataset parser inside the *instance* package and defines the specific calculus syntax inside the *objective* package. The remaining communication flow and execution sequence are handled by the core abstractions.

(2) Adaptation Module: Employs polymorphism and inheritance to inject interactivity into existing SBSE or iSBSE tools. By extending the framework's core classes, a search tool can be retrofitted with interactive preference capture and ML-based evaluation.

3.3 Graphical Interface

Figure 2 shows the initial page of the *ISearchAI* graphical interface, in which the software engineer can develop interactive approaches with the framework.

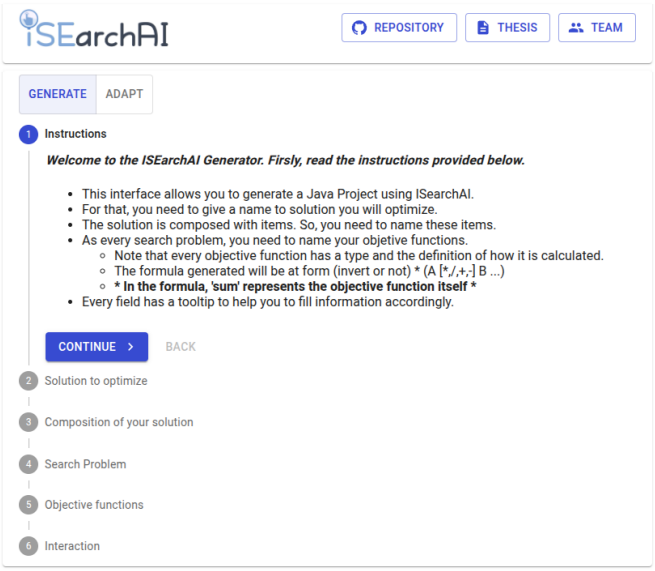

Figure 2: *ISearchAI* Graphical Interface

The two main buttons available in the framework ("Generate" and "Adapt") are used to generate and adapt approaches. When the button to generate the iSBSE approaches is clicked, the software engineer first receives the necessary instructions to proceed with the generation process ("1 - Instructions"). When moving to the second stage ("2 - Solution to optimize"), the engineer defines the solution to be optimized. Then, the composition of the solution is defined ("3 - Composition of your solution").

In the definition of the search problem (*4 - Search Problem*), the encoding of the approach's input solution is specified as binary, integer, or double, and the search algorithm is specified as well. In step 5 (*Objective functions*), the objective functions are defined, as well as their calculation. Finally, in step 6 (*Interaction*), the interaction with the DM will be defined. The adaptation part is not shown

in the figure, as it follows a pattern similar to that of the generation, but examples are presented in the supplementary material [16].

4 Experimental Design

This section details the evaluation design conducted to assess the technical feasibility of the *ISearchAI* framework. The framework's evaluation comprises two experiments: Experiment 1 and Experiment 2. Figure 3 outlines the operational steps, variables, and data collection procedures established independently for each experiment. The initial step of both experiments involved installing and configuring the framework. The experimental data, analysis scripts, and source code for the *ISearchAI* can be found in [16].

Experiment 1 was conducted to evaluate Case Study 1, an instance of Workflow 1 (generation workflow), in which software engineers and researchers (the participants in this experiment) used the *ISearchAI* framework to develop new iSBSE approaches for the NRP domain. Conversely, **Experiment 2** addressed Case Study 2, an instance of Workflow 2 (adaptation workflow), which involved adapting OPLA-Tool v2.0 [19], an established tool for interactive PLA optimization.

The selection of the NRP (Experiment 1), in conjunction with the PLA domain (Experiment 2), provides a diverse experimental baseline for evaluating domain independence. This variety ensures that the architecture remains functional across different search landscapes and solution encodings, regardless of structural complexity.

The resulting implementations and the optimized solutions provided to the participants of the experiments were then instrumented for data collection. Finally, a quantitative analysis was performed to assess the framework's performance in supporting both generation and adaptation workflows. The evaluation strictly assesses technical feasibility, functional correctness, and architectural non-intrusiveness; a formal evaluation of human development effort remains out of scope for this study.

Based on the definition of the research proposal and the primary objective of this work, the following Research Questions (RQs) were formulated to evaluate the framework's functional capabilities:

RQ1 (Case Study 1): To what extent does the *ISearchAI* framework enable the construction of functional iSBSE approaches by software engineers and researchers with varying levels of experience? This question evaluates Workflow 1 (generation) by investigating whether the framework's infrastructure allows participants to successfully transition from conceptual definitions to executable search approaches that effectively optimize domain-specific objectives. Experiment 1 was performed to answer this question.

RQ2 (Case Study 2): To what extent does the architectural adaptation of established tools via *ISearchAI* preserve the functional performance and search integrity of the original approaches? This question focuses on Workflow 2 (adaptation) by using statistical analysis to verify whether retrofitting existing systems, such as the OPLA-Tool v2.0, maintains the original search performance without introducing negative artifacts. Experiment 2 was performed to answer this question.

The experimental foundation of these studies comprises two representative problems from the SBSE literature. The NRP is presented as a combinatorial optimization challenge focused on requirement selection under cost constraints [3]. Additionally, the PLA domain is explored using the OPLA-Tool v2.0 [19], which automates the iMOA4PLA interactive approach [19]. The OPLA-Tool evaluation uses architectural design metrics to compute the optimization fitness, providing the metrics for evaluating search trajectories under interactive constraints [20]. In this work, we used specifically Feature Modularization (FM), Class Coupling (ACLASS), and Cohesion (COE) [35].

Table 1 describes the technical configurations used in the experiments. These configurations are aligned with the original works that proposed the base approaches used for validation [11, 20]. Based on these settings, interactive iSBSE approaches were implemented and evaluated across two primary case studies. Both experiments were conducted in a standardized hardware environment to ensure consistency in execution time and resource consumption data.

Table 1: Experimental Environment Configuration

Component	Specification
Hardware	
CPU	Intel Core i7
RAM Memory	16 GB DDR4
Operating System	Windows 11 / macOS / Linux
Software	
Java Development Kit (JDK)	Version 11.X
Maven	Version 3.8.X
IDE	IntelliJ IDEA
Evaluated Framework	*ISearchAI* (Version 1.0)
Experiment 1 - Generation Workflow (NRP Scenario based on [11])	
Base Problem	NRP optimization
Data Set	100 requirements
Instrument	Objective function analysis
Problem Type	Floating Point *Double*
Search Algorithm	NSGA-II / NSGA-III
Objective Functions	Cost, Import, Profit, and Size
ML Algorithm	Multilayer Perceptron (MLP)
Experiment 2 - Adaptation Workflow (PLA Scenario based on [20])	
Base Problem	PLA optimization
Dataset	PLA AGM
Instrument	Statistical analysis
Problem Type	Object-oriented
Search Algorithm	NSGA-II
Objective Functions	COE, ACLASS, and FM
ML Algorithm	Multilayer Perceptron (MLP)

4.1 Experiment 1

As illustrated in Figure 3, Experiment 1 was designed as a controlled user experiment performed through seven sequential operational steps to evaluate the Generation Workflow (Workflow 1). To address RQ1, the experiment treats the participant experience level as the independent variable and the functionality of the generated code as the dependent variable.

1) Framework setup. The experiment initiated with the installation and configuration of the *ISearchAI* framework. This step

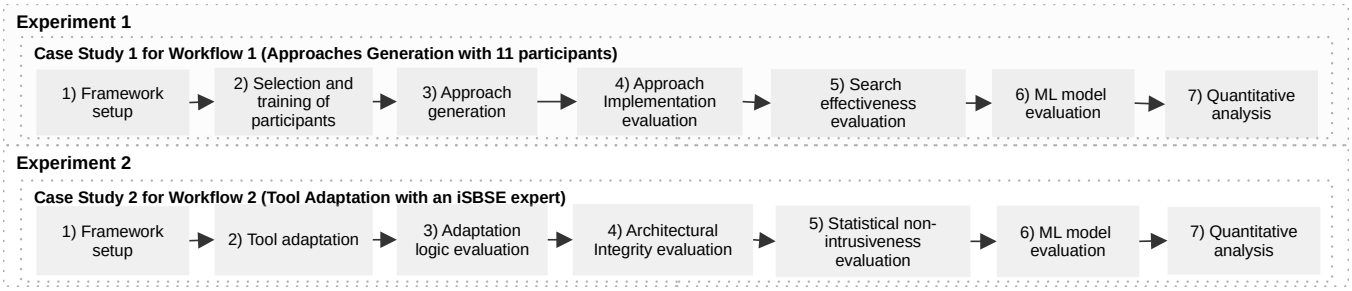

Figure 3: Experimental Design

established the standardized environment required to assess the framework's capacity to automate the development of new iSBSE approaches and ensured that all foundational dependencies were correctly instrumented for data collection.

2) Selection and training of participants. This step involved the recruitment of 11 participants, comprising software engineers and researchers. The selection was designed to represent practitioners focused on software construction and researchers focused on experimental prototyping. Participants underwent a training protocol to align their domain knowledge of the NRP and the framework's operational mechanics, thereby mitigating technical barriers related to the tool's interface.

3) Approach generation. Here, participants executed optimization tasks using the framework's capabilities to rapidly construct functional multi and many-objective approaches from high-level problem definitions. This step focused on the transition from conceptual problem specifications to executable search logic, producing the optimization artifacts that were subsequently analyzed.

4) Approach Implementation evaluation. The resulting implementations were instrumented and reviewed. The objective functions proposed by participants were analyzed; however, those relying on arbitrary or poorly substantiated definitions were excluded from the quantitative comparison to maintain the evaluation's validity. This ensured that the assessed artifacts reached the technical standard required for a robust search process. In this sense, only the objective functions Cost, Importance, and Profit were analyzed.

5) Search effectiveness evaluation. The optimized solutions were assessed to verify the functional effectiveness of the generated approaches. The results were analyzed using standardized objective functions (cost, importance, and profit), enabling comparisons with the original average requirement values obtained in [12].

6) ML model evaluation. Because the approaches integrate ML components to simulate DM preferences, participants executed their generated models while acting as DMs. They provided training data (scores from 0 to 100) in predefined iterations. The ML models, specifically utilizing the Multilayer Perceptron (MLP) algorithm [28], were evaluated for accuracy and agreement using the Kappa coefficient [7]. These indicators assessed whether the models effectively learned the DM's profile, although the focus remained on the framework's ability to support such integration rather than the ML algorithms themselves.

7) Quantitative analysis. The final step involved the consolidation and analysis of the collected results, performed by an expert in iSBSE. This analysis assessed the framework's capacity to automate the development process and enabled the drawing of lessons regarding the role of domain knowledge.

Table 2: Profile of the participants in Experiment 1.

ID	Role / Education	Java Exp.	SBSE Knowledge
P01	Researcher / Bachelor	Basic	Basic
P02	Researcher / Bachelor	Intermediate	Intermediate
P03	Engineer / Bachelor	Intermediate	Intermediate
P04	Researcher / Bachelor	Basic	Basic
P05	Engineer / Bachelor	Intermediate	Intermediate
P06	Engineer / Bachelor	Basic	Basic
P07	Engineer / Bachelor	Basic	Intermediate
P08	Researcher / Master	Basic	Basic
P09	Engineer / Master	Basic	Intermediate
P10	Researcher / Bachelor	Basic	Basic
P11	Expert / Master	Intermediate	Proficient

4.1.1 Participants Profile. The selection of participants was designed to represent the two primary user groups of the *ISearchAI* framework: practitioners focused on software construction, and researchers focused on experimental prototyping. Participants were tasked to evaluate the framework's capacity to support practical development tasks, while researchers provided insight into its utility for academic investigation. By including individuals with varying levels of experience in Java and SBSE, the study assesses whether the framework effectively abstracts implementation complexities for non-specialists while remaining a functional tool. This diversity ensures that the evaluation captures the framework's accessibility and its effectiveness in mitigating technical barriers across different levels of proficiency within the software engineering community.

The group's diversity enabled evaluation of the framework across levels of technical proficiency, from undergraduate students to PhD candidates and industry professionals. Table 2 summarizes the demographic and technical characteristics of the participants, detailing their roles, education, and prior knowledge of the technologies utilized in the work. All participants completed the tasks related to generating iSBSE approaches within a standardized timeframe.

4.2 Experiment 2

As illustrated in Figure 3, in Experiment 2, Case Study 2 was structured as a seven-step operational sequence to evaluate the Adaptation Workflow (Workflow 2) in a realistic legacy system context. This case study aimed to answer RQ2 by assessing whether the integration of the *ISearchAI* layered architecture preserves the search integrity and performance distributions of the target consolidated tool. The process followed the methodology described below:

1) Framework setup. Similar to the previous experiment, this step involved the initialization and configuration of the *ISearchAI* environment. This established a baseline for the adaptation process, ensuring proper instrumentation of the framework dependencies to monitor integration with a complex source code base.

2) Tool adaptation. The core of this step involved the integration of the framework into OPLA-Tool v2.0. *ISearchAI* was included as an external dependency, requiring adaptations to specific internal classes to support the iMOA4PLA approach. This step tested the flexibility of the hybrid architecture when integrated into a non-trivial legacy system without requiring a fundamental rewrite of the original optimization logic.

3) Adaptation logic evaluation. This step focused on verifying the mapping between the legacy search logic and the framework's abstraction layer. It was ensured that the architectural mutation operators and evaluation metrics were correctly integrated into the *ISearchAI* search process, minimizing implementation errors stemming from an incomplete understanding of the OPLA-Tool's internal architecture.

4) Architectural Integrity evaluation. The study verified whether incorporating the framework preserved the integrity of the original optimization constraints. This involved assessing the persistence of DM preferences, such as frozen architectural elements, within the optimized solutions. The goal was to ensure that the framework facilitated interactive constraints without introducing artifacts that could misalign the results with the DM's profile.

5) Statistical non-intrusiveness evaluation. Technical feasibility was measured through a non-inferiority analysis between the original version (denoted as **OPLA-Tool-1**) and the adapted version (denoted as **OPLA-Tool-2**). Both versions were executed in accordance with the Bindewald et al. [5] protocol. Since the Shapiro-Wilk test indicated a non-normal data distribution ($p < 0.05$), the non-parametric Kruskal-Wallis test (95% confidence level) was applied. The success criterion was the absence of statistically significant differences in the objective function values (COE, ACLASS, and FM), validating performance neutrality.

6) ML model evaluation. The performance of the integrated ML models, which simulated DM preferences via the MLP algorithm, was assessed. The DM provided training data (scores and freezes) over a predefined number of iterations. Accuracy was calculated to assess performance relative to human evaluations, complemented by the Kappa coefficient [7] to evaluate agreement beyond chance. This confirmed the architecture's ability to maintain high predictive fidelity even in high-granularity architectural design tasks.

7) Quantitative analysis. The final step consolidated the statistical and predictive data to confirm the framework's effectiveness in supporting adaptation workflows. This analysis provided evidence that *ISearchAI* can be integrated into consolidated codebases while preserving search performance and integrity, thereby ensuring the case study's technical validity.

4.2.1 Participant Profile. In contrast to the participant group in Experiment 1, the adaptation evaluation in Experiment 2 was conducted by an expert with a PhD in Computer Science specializing in iSBSE. This is motivated by the nature of the experiment, which aims to evaluate architectural feasibility and non-intrusiveness. Given the technical complexity of OPLA-Tool and the requirements for adapting a legacy system, a domain expert was engaged to ensure that legacy logic was mapped onto the framework's abstraction layer. This protocol was adopted to minimize implementation errors arising from an incomplete understanding of the OPLA-Tool's internal architecture, thereby ensuring the validity of the case study evaluation.

The decision to use an expert-driven evaluation at this step does not preclude future studies involving external participants. Establishing the technical viability of the adaptation module is a requirement before conducting generalizability tests. Due to the domain knowledge required to navigate the OPLA-Tool's architectural patterns, external participants were excluded from the scope of the current work.

5 Results

The evaluation of the *ISearchAI* framework was conducted through two case studies, focusing on the primary workflows of automated generation and adaptation of existing SBSE approaches. Subsection 5.1 presents the technical outcomes of the search approach generated for the NRP, while Subsection 5.2 details the findings from retrofitting OPLA-Tool for PLA design. Finally, Subsection 5.3 presents answers to RQs.

5.1 Experiment 1

The 11 participants in this experiment were asked to develop approaches to the NRP problem using the *ISearchAI* framework. A default configuration (presented in Table 1) was provided to assist participants, but they were free to modify details such as the objective functions, the name, and the composition of the solution to be optimized. These configurations were based on the work of [11].

The baseline objectives comprise cost and size (to be minimized), alongside importance and profit (to be maximized), following the problem configurations established in [11]. These objective functions were presented as the default to the participants, with the option to modify them.

The diversity in the approaches generated by participants demonstrated the framework's flexibility. While a default configuration was provided, over 50% of participants used the framework's extensibility to modify or remove the Size objective function. It is worth noting that all participants chose to maintain the objective functions Cost, Importance, and Profit as the original and the input solution with 100 requirements, provided by [11]. In this sense, to maintain consistency in the quantitative analysis, we evaluated the objective functions Cost, Importance, and Profit separately from the others, as they remain uniform across the generated functional approaches.

In this experiment, the Size objective function was modified by some participants. When analyzing this specific objective function,

we observed that it added no value to the search (neither positive nor negative), primarily because participants defined it randomly. As the objective of this experiment is not to find the optimal solution for the NRP, but rather to validate whether *ISearchAI* is useful for generating new approaches and adapting existing tools, the following analysis focused on the Cost, Importance, and Profit.

Table 3 presents the averages of the objective function values obtained from the original work [12] for the requirements used in the experiments. To evaluate the effects of the generated search approaches, the normalized averages of the original values (Table 3) were compared with those obtained from the resulting optimized solutions. Table 3 summarizes this comparison. It presents the objective functions' Cost, Importance, and Profit, with their original values from the input solution and the average values across the optimized solutions. As shown in this table, the objectives that improved were Cost and Importance.

Table 3: Normalized means (Original vs. Generated)

Objectives	Original work [12]	Generated
Cost	0.50	0.43
Importance	0.46	0.48
Profit	0.53	0.48

The generated approaches yielded solutions with lower average cost and higher average importance, indicating a prioritization of lower-cost, high-priority requirements, while average profit declined as an optimization trade-off.

The observed reduction in average profit, alongside improvements in cost and importance, reflects a typical trade-off, indicating that the approaches generated via *ISearchAI* operationalized the underlying metaheuristic to explore the search space, prioritizing implementability and requirements over potential returns.

As discussed earlier, the participants in the experiments defined distinct objective functions, including modifications to the original formulas [12] used to compute them. When analyzing these functions, we observed some randomness in several definitions, mainly due to participants' limited knowledge of the problem. Furthermore, the short experimental duration made it difficult for participants to define the objective functions, and some poorly defined ones affected the search process.

Analyzing the results in Table 3, we observe that the solutions were optimized, resulting in limited numerical variation across the optimization goals. This output is directly linked to the objective definitions supplied by non-expert participants, who occasionally formulated constraints with a limited optimization trajectory. This constraint highlights that automating infrastructure steps does not eliminate the core requirement for modeling analytical problems.

These results show that while the framework abstracts the instrumental complexity of iSBSE, it preserves the participant's role in domain modeling. The observed performance variations underscore the necessity of domain expertise in defining objective functions, a task that *ISearchAI* facilitates through its layered architecture without imposing rigid constraints on the problem's logic.

However, based on the averages in Table 3, the approaches generally slightly improved the solutions, despite these limitations. A relevant qualitative aspect of this experiment, in addition to the numerical values of the objective functions, is the development time. Participants in this study completed the structural configuration and generated the template files within a single session, restricted to one hour. While this setup demonstrates the functional execution of the generation workflow within a designated timeframe, it does not constitute a comparative evaluation of development effort relative to a manual implementation baseline.

The focus of this work was the evaluation of the technical feasibility of generated approaches using the *ISearchAI* framework. Thus, quality indicators for multiobjective optimization, including Hypervolume, were omitted from this technical evaluation step. Because the participants produced distinct, non-standardized mathematical formulas for the objective functions, a centralized Pareto-front indicator could not be calculated. The verification focused strictly on the workflow's instantiation mechanics. Specifically, the inclusion of indicators such as Hypervolume will enable a quantitative assessment of the quality and diversity of the results, enabling a formal performance comparison between the approaches generated via *ISearchAI* and established non-interactive baselines.

Regarding the expert's specific approach, similarities were observed in the definitions of objective functions among non-expert participants. They had similar issues with their definitions. When evaluating the generated approach, we observed that the search results were close to those obtained by other participants. This indicated that **even non-iSBSE specialists can use the *ISearchAI* framework in a manner similar to that of specialists.**

The optimized solutions obtained by executing each approach were collected and analyzed. To mitigate the cognitive stress associated with excessive interactions, ML-based preference models were integrated into the search process alongside the participants. These models, analogous to those employed in established iSBSE environments such as the OPLA-Tool, are trained during interactions by learning preference patterns from solution data, such as the numerical scores assigned by the DM. After a predefined number of interactions, the ML model assumes the role of the participant and autonomously evaluates the optimized solutions to sustain the search process without further human intervention.

The predictive models were trained using dataset instances gathered across three interactive synchronization rounds, with each participant evaluating a set of 20 solutions per round to establish preference labels. The feature representation maps the binary solution vector of requirement selections into discrete numerical scores while preserving the native distribution without artificial balancing. To prevent data leakage and ensure structural validity, a 10-fold cross-validation mechanism was performed independently for each user profile during the offline training cycle [23]. All decisions related to ML algorithms were made based on the approaches/tools original works used in the experiments [5, 12, 20].

The predictive accuracy of the classifier was measured during the execution phase by evaluating its classification consensus against a held-out testing partition. The ground truth was defined as the explicit preference scores that participants manually assigned to a subset of solutions during the synchronized interaction rounds.

Figure 4 summarizes the predictive accuracy across participant models. As a sanity check to verify whether the classifier fits individual evaluation patterns over small per-profile datasets (60

instances), the MLP model reached a mean accuracy of 96.86% with a Kappa coefficient [7] exceeding 0.80. This confirms that the framework correctly transfers human preference labels into the automated classification loop.

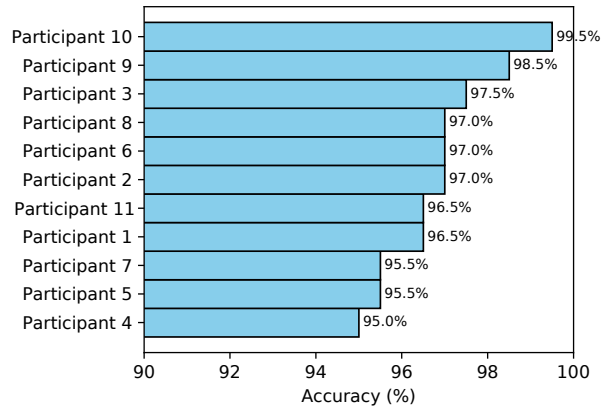

Figure 4: ML models accuracy for each participant

The evaluation of the ML models' performance demonstrated their ability to reproduce the participants' preferences. It was found that the model evaluated the optimized solutions similarly to the participants, and that 98.0% of the participants' preferences were present in the final solutions. That is, 98% of the desired requirements were in the resulting solutions. This indicates that the model learned the participant's profile, preventing the algorithm from generating solutions misaligned with their preferences.

5.2 Experiment 2

Since the *ISearchAI* framework can be incorporated into existing iSBSE approaches, we evaluated the adaptation of iSBSE to *ISearchAI* by implementing the framework in OPLA-Tool. In this approach, the framework was included as an external dependency, and some classes were adapted to support it. OPLA-Tool was executed twice: once with the framework and once without it. To organize the results, the version without the framework was named OPLA-Tool-1, and the version with the framework was OPLA-Tool-2. Both approaches were implemented by an iSBSE expert, who adapted the tool and engaged in the search process.

In this study, OPLA-Tool-1 serves as a baseline execution utilizing a customized non-interactive automation layer of the OPLA-Tool infrastructure. This setup allows the evaluation to measure the structural impact of embedding the *ISearchAI* interactive components into an automated optimization baseline, denoted as OPLA-Tool-2.

The evaluation confirmed that the DM's preferences were reflected in the final results, as evidenced by the persistence of frozen classes in the optimized solutions. By identifying specific architectural elements that remained unchanged throughout the search process, the study demonstrated that **incorporating the framework into OPLA-Tool preserved the integrity of the original optimization logic**. Furthermore, the search algorithm yielded results consistent with previous experimental protocols, indicating that the framework facilitates interactive constraints without introducing negative artifacts into the established search process.

At the end of the search for the two tools, the optimized solutions and objective function values were collected, and the statistical significance of the difference between the results obtained with OPLA-Tool-1 and OPLA-Tool-2 was assessed. All tests were performed with a 95% confidence level. Before analyzing statistical differences, the Shapiro-Wilk test was applied to assess the normality of the data. The test indicated that neither OPLA-Tool-1 nor OPLA-Tool-2 presented a normal distribution (p-value < 0.05).

Given the data distribution, the Kruskal-Wallis test was applied to assess statistically significant differences between OPLA-Tool-1 and OPLA-Tool-2. This test showed no statistically significant difference between the results obtained with both approaches. The lack of a statistically significant difference ($p > 0.05$) indicates that the collected optimization metrics do not present sufficient evidence to reject the null hypothesis of identical distributions. Although this non-parametric test does not mathematically prove absolute equivalence or non-inferiority within specified boundaries, the high similarity among the objective values indicates that integrating the framework did not cause functional anomalies in the search path.

As summarized in Table 4, OPLA-Tool-1 achieved objective values similar to OPLA-Tool-2. This close proximity in objective scores, combined with the lack of statistical variance ($p > 0.05$), indicates that the framework abstraction layers can capture interactive preferences and manage machine learning instances without causing systematic functional deviations in the baseline search execution.

Table 4: Objective function values for the best solutions (Original vs. Adapted)

Objectives	OPLA-Tool-1	OPLA-Tool-2
FM	684	685
ACLASS	12	12
COE	32	31

It is worth noting that *ISearchAI's* goal is not to improve the solutions produced by the interactive search process, but rather to assist *ISearchAI* users in implementing iSBSE approaches. We summarized the evaluation to determine whether there was a negative impact on an already consolidated interactive search approach.

Integrating *ISearchAI* into OPLA-Tool-2 enabled two interaction mechanisms: global solution scoring (on a 0–5 quality scale) and freezing of architectural elements based on user preferences.

The average accuracy of the OPLA-Tool-2 ML models was measured as the percentage of correct classifications, as in Experiment 1. An average accuracy of 80.0% was observed, and the Kappa coefficient [7] exceeded 0.77 in the last iteration, indicating substantial agreement and confirming the reliability of the training data.

In this sense, the evaluation of the OPLA-Tool-2 ML models performance demonstrated its ability to replicate the DM's preferences. We have also found that the model froze elements similar to those chosen by the DMs; 98.0% of the elements the DMs froze were present in the solutions at the end of the process. This indicates that the model learned the DM's profile, thereby preventing the generation of solutions misaligned with the DM's preferences.

5.3 Answers to RQs

This subsection presents the answers to the RQs, based on results obtained from the case studies performed in this work.

RQ1: To what extent does the *ISearchAI* framework enable the construction of functional iSBSE approaches by software engineers and researchers with varying levels of experience?

The outcomes of Experiment 1 demonstrate that *ISearchAI* enables the construction of functional iSBSE approaches by abstracting the instrumental complexity of the search infrastructure. All 11 participants, representing diverse technical backgrounds, transitioned from high-level conceptual definitions to executable search logic. The functional viability of these approaches is evidenced by the optimization trends observed in the NRP domain, where participants generated solutions that yielded improved normalized means for cost and importance. Furthermore, the 96.86% accuracy of ML models confirms that the framework's Generation Workflow allows software engineers and researchers to successfully operationalize human preference modeling within the search process, regardless of their prior expertise in SBSE or iSBSE.

RQ2: To what extent does the architectural adaptation of established tools via *ISearchAI* preserve the functional performance and search integrity of the original approaches?

Statistical validation through Experiment 2 demonstrates that the framework preserves search integrity. The Kruskal-Wallis test showed no performance degradation after the OPLA-Tool's adaptation, confirming the framework's architectural transparency. Furthermore, the 98% persistence of frozen architectural elements confirms that the framework's interaction module effectively operationalizes human constraints within the search process without introducing negative artifacts or degrading mathematical integrity.

6 Threats to Validity

This section discusses the potential limitations of the evaluation.

Internal Validity: This threat concerns internal factors that could affect the observed results. To ensure the reliability of the integrated ML models and to avoid overfitting, a 10-fold cross-validation protocol was used during training. In Experiment 2, the stochastic nature of the algorithms was mitigated by conducting 15 independent executions, ensuring that the comparison was based on a representative distribution of the search process. Case Study 2 relied on a single domain expert to complete the tool adaptation. Although this configuration is well-suited for evaluating architectural feasibility and backend coupling in legacy architectures, it limits the generalizability of claims about the workflow's broad usability and cognitive accessibility across different profiles.

External Validity: The primary threat to the results' generalization is the limited number of problem domains analyzed. While the NRP and PLA optimization represent distinct SBSE challenges, ranging from requirement prioritization to architectural design, the results may vary when the framework is applied to other software engineering tasks. [13]. However, the diversity of the encodings suggests potential for applicability in other iSBSE contexts.

Construct Validity: This threat relates to whether the evaluation metrics accurately reflect the framework's performance goals. We addressed this by utilizing standard multiobjective optimization indicators and domain-specific architectural metrics to measure performance. Using statistical non-inferiority as a proxy for architectural non-intrusiveness ensures that the framework's abstraction layer does not affect the approach's fundamental search logic.

Conclusion Validity: To ensure the statistical significance of the findings, data normality was verified through Shapiro-Wilk test. Given the non-normal distribution of the results, the non-parametric Kruskal-Wallis test was applied to compare the adapted and original tools [18]. This ensures that the conclusion regarding the framework's non-intrusiveness is mathematically grounded rather than due to random variation in the metaheuristic search.

7 Conclusion

The implementation of iSBSE approaches, particularly those employing ML to reduce cognitive load, has encountered technical challenges. These limitations are attributed to the absence of standardized infrastructures capable of integrating evolutionary search with real-time preference modeling [31]. This research addresses these gaps by proposing and validating *ISearchAI*, a framework that facilitates both the generation and adaptation of iSBSE tools.

The technical performance of the framework was evaluated through two case studies focusing on core architectural workflows. The development of an iSBSE approach for the NRP demonstrated the framework's capability to automate the construction of requirement-prioritization methods. Results indicated that *ISearchAI* reduces the implementation effort for iSBSE processes, as the integrated ML models achieved predictive fidelity with a mean accuracy of 96.86% and a Cohen's Kappa coefficient exceeding 0.80. This indicates that standardizing the iSBSE development lifecycle mitigates technical constraints associated with these approaches [24].

Case Study 2 provided evidence of architectural non-intrusiveness. Applying the adaptation workflow to a pre-existing tool revealed statistical similarity between the indicators. The data distribution indicated that the baseline search execution was maintained during the integration tests. By abstracting data transformations and providing a unified interface for ML-assisted search, the framework allows researchers to focus on preference modeling and decision-making [14]. The ability to generate approaches and adapt tools establishes *ISearchAI* as a research framework that supports human-centered search within the software engineering community.

Future work will focus on expanding the evaluation module to include automated reporting for search metrics and extending support to additional software domains [1]. Additionally, we plan to integrate Large Language Models (LLMs) to assist DMs. We are also currently evaluating the framework's usability. The availability of an open-source, standardized framework is expected to support scientific reproducibility and the development of hybrid approaches that combine human expertise with automated metaheuristic search.

Artifact Availability

The artifacts produced in this study are publicly available at https://doi.org/10.6084/m9.figshare.32691540.

Acknowledgments

The authors thank CAPES Funding Code 001 and CNPq (Grants 404027/2023-7 and 306774/2025-9) for financial support.

References

[1] Caio Arasaki, Lucas Wolschick, Willian Freire, and Aline Amaral. 2023. Feature selection in an interactive search-based PLA design approach. In *Proceedings of the 17th Brazilian Symposium on Software Components, Architectures, and Reuse*. 11–20.

[2] Allysson Allex Araújo and Matheus Paixão. 2014. Machine learning for user modeling in an interactive genetic algorithm for the next release problem. In *International Symposium on Search Based Software Engineering*. Springer, 228–233.

[3] Bagnall, Rayward-Smith, and Whittley. 2001. The next release problem. *Information and Software Technology* 43, 14 (2001), 883–890. doi:10.1016/s0950-5849(01)00194-x

[4] Carlos Vinicius Bindewald, Willian M Freire, Aline MM Miotto Amaral, and Thelma Elita Colanzi. 2019. Towards the support of user preferences in search-based product line architecture design: an exploratory study. In *Proceedings of the XXXIII Brazilian Symposium on Software Engineering*. 387–396.

[5] Carlos Vinicius Bindewald, Willian M Freire, Aline MM Miotto Amaral, and Thelma Elita Colanzi. 2020. Supporting user preferences in search-based product line architecture design using Machine Learning. In *Proceedings of the 14th Brazilian Symposium on Software Components, Architectures, and Reuse*. 11–20.

[6] Samuel Bleuler, Marco Laumanns, Lothar Thiele, and Eckart Zitzler. 2003. PISA – A platform and programming language independent interface for search algorithms. In *Proceedings of the 2nd International Conference on Evolutionary Multi-Criterion Optimization (EMO)*. Springer-Verlag, Faro, Portugal, 494–508.

[7] Jacob Cohen. 1960. A coefficient of agreement for nominal scales. *Educational and psychological measurement (EPM)* 20 (1960), 37–46. doi:10.1177/001316446002000104

[8] Thelma Elita Colanzi, Silvia Regina Vergilio, Itana M. S. Gimenes, and Willian Nalepa Oizumi. 2014. A Search-Based Approach for Software Product Line Design. In *18th International Software Product Line Conference (SPLC)*. 237–241.

[9] Juan J Durillo and Antonio J Nebro. 2011. jMetal: A Java framework for multi-objective optimization. *Advances in Engineering Software* 42, 10 (2011), 760–771.

[10] Thiago Ferreira, Silvia Regina Vergilio, and Marouane Kessentini. 2020. Nautilus: An Interactive Plug-and-Play Search-Based Software Engineering Framework. *IEEE Software* 38, 5 (2020), 73–82.

[11] Thiago do Nascimento Ferreira. 2019. *A preference-based approach for reducing the number of objectives applied to the variability testing of software product line*. Ph. D. Dissertation. Programa de Pós-Graduação em Informática, Universidade Federal do Paraná, Curitiba, PR. https://acervodigital.ufpr.br/xmlui/handle/1884/65576 Acesso em: 07 abr. 2025.

[12] Thiago do Nascimento Ferreira, Silvia Regina Vergilio, and Marouane Kessentini. 2021. Implementing Search-Based Software Engineering Approaches with Nautilus. In *Proceedings of the XXXV Brazilian Symposium on Software Engineering*. 303–308.

[13] Thiago Nascimento Ferreira, Silvia Regina Vergilio, and Jerffeson Teixeira de Souza. 2017. Incorporating user preferences in search-based software engineering: A systematic mapping study. *Information and Software Technology* 90 (2017), 55–69.

[14] Claes Fornell and David F Larcker. 1981. Evaluating structural equation models with unobservable variables and measurement error. *Journal of marketing research* 18, 1 (1981), 39–50.

[15] Félix-Antoine Fortin, François-Michel De Rainville, Marc-André Gardner, Marc Parizeau, and Christian Gagné. 2012. DEAP: Evolutionary algorithms made easy. *The Journal of Machine Learning Research* 13, 1 (2012), 2171–2175.

[16] Willian Freire, Aline Maria Malachini Miotto Amaral, and Thelma Elita Colanzi. 2026. ISearchAI: A Hybrid Framework for Streamlining Interactive Search-Based Software Engineering with Machine Learning. doi:10.6084/m9.figshare.32691540

[17] Willian Freire, Cláudia Rosa, Aline Amaral, and Thelma Colanzi. 2022. Validating an interactive ranking operator for NSGA-II to support the optimization of software engineering problems. In *Proceedings of the XXXVI Brazilian Symposium on Software Engineering*. 337–346.

[18] Willian Freire, Simone Tonhão, Tiago Bonetti, Marcelo Shigenaga, William Cadette, Fernando Felizardo, Aline Amaral, Edson OliveiraJr, and Thelma Colanzi. 2021. On the configuration of multi-objective evolutionary algorithms for PLA design optimization. In *15th Brazilian Symposium on Software Components, Architectures, and Reuse*. 11–20.

[19] Willian Marques Freire, Mamoru Massago, Cattaneo Arthur Zavadski, Aline MM Miotto Amaral, and Thelma Elita Colanzi. 2020. OPLA-Tool v2.0: a Tool for Product Line Architecture Design Optimization. In *34th Brazilian Symposium on Software Engineering (SBES '20), October 21–23, 2020, Natal, Brazil*. ACM.

[20] Willian Marques Freire, Cláudia Tupan Rosa, Aline Maria Malachini Miotto Amaral, and Thelma Elita Colanzi. 2024. Interactive search-based Product Line Architecture design. *Automated Software Engineering* 31, 2 (2024), 1–39.

[21] David Hadka. 2014. MOEA Framework: A Free and Open Source Java Framework for Multiobjective Optimization. http://moeaframework.org/

[22] Mark Harman, S Afshin Mansouri, and Yuanyuan Zhang. 2012. Search-based software engineering: Trends, techniques and applications. *ACM Computing Surveys (CSUR)* 45 (2012), 11. doi:10.1145/2379776.2379787

[23] Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani. 2023. *An Introduction to Statistical Learning: with Applications in R* (2nd ed.). Springer Nature, New York. doi:10.1007/978-1-0716-1418-1

[24] Rensis Likert. 1932. A technique for the measurement of attitudes. *Archives of psychology* (1932).

[25] Giovanni Misitano, Bhupinder Singh Saini, Bekir Afsar, Babooshka Shavazipour, and Kaisa Miettinen. 2021. DESDEO: The modular and open source framework for interactive multiobjective optimization. *IEEE Access* 9 (2021), 148277–148295.

[26] Aurora Ramirez, Jose Raul Romero, and Christopher Simons. 2018. A systematic review of interaction in search-based software engineering. *IEEE Transactions on Software Engineering* (2018).

[27] Aurora Ramirez, José Raúl Romero, and Sebastian Ventura. 2018. Interactive multi-objective evolutionary optimization of software architectures. *Information Sciences* 463 (2018), 92–109.

[28] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. 1986. Learning representations by back-propagating errors. *nature* 323, 6088 (1986), 533–536.

[29] Eric O Scott and Sean Luke. 2019. *ECJ at 20: Toward a general metaheuristics toolkit*. Technical Report. 1391–1398 pages.

[30] Mark Shackelford. 2007. Implementation issues for an interactive evolutionary computation system. In *Proceedings of the 9th annual conference companion on Genetic and Evolutionary Computation*. ACM, 2933–2936.

[31] Chris Simons and Jim Smith. 2016. Exploiting antipheromone in ant colony optimisation for interactive search-based software design and refactoring. In *Proceedings of the 2016 on Genetic and Evolutionary Computation Conference Companion*. ACM, 143–144.

[32] Hideyuki Takagi. 2002. Interactive evolutionary computation: Fusion of the capabilities of EC optimization and human evaluation. *Proc. IEEE* 89, 9 (2002), 1275–1296.

[33] The Apache Software Foundation. 2026. *Apache Maven Project*. https://maven.apache.org

[34] Ye Tian, Ran Cheng, Xingyi Zhang, and Yaochu Jin. 2017. PlatEMO: A MATLAB platform for evolutionary multi-objective optimization [educational forum]. *IEEE Computational Intelligence Magazine* 12, 4 (2017), 73–87.

[35] Yenisei Delgado Verdecia, Thelma Elita Colanzi, Silvia Regina Vergilio, and Marcelo C. Santos. 2017. An Enhanced Evaluation Model for Search-based Product Line Architecture Design. In *20th Iberoamerican Conference on Software Engineering (CIbSE)*. CIbSE, San Jose, Costa Rica, 155–168.

[36] Claes Wohlin, Per Runeson, Martin Höst, Magnus C Ohlsson, Björn Regnell, Anders Wesslén, et al. 2012. *Experimentation in software engineering*. Vol. 236. Springer.