AIssistDM: A Plugin to Assist Non-specialist Decision-Makers in Search-Based Software Engineering Tools

Willian M. Freire
State University of Maringá
Maringá, Paraná, Brazil
willianmarquesfreire@gmail.com

Murilo Boccardo
State University of Maringá
Maringá, Paraná, Brazil
ra124160@uem.br

Daniel Nouchi
State University of Maringá
Maringá, Paraná, Brazil
ra123991@uem.br

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

Silvia R. Vergilio
DInf, Federal University of Paraná
Curitiba, Paraná, Brazil
silvia@inf.ufpr.br

Thiago Ferreira
University of Michigan-Flint
Flint, MI, USA
thiagod@umich.edu

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

ABSTRACT

Search-Based Software Engineering (SBSE) is a field that applies optimization algorithms to address complex problems in various software engineering domains. A significant challenge when using SBSE tools is the choice of parameter values related to the problem domain, which is usually performed by *the Decision-Makers (DMs).* This task is not trivial and critically influences the resulting solutions. Recognizing that DMs, mainly non-specialist ones, often struggle with these complexities, this paper introduces AIssistDM, a plugin developed to make adopting SBSE tools accessible without needing deep knowledge of their configuration parameters. The AIssistDM offers guided assistance by integrating *Large Language Models (LLMs)*, particularly ChatGPT, to provide human-text generated suggestions of problem-domain parameters, improving decision-making. AIssistDM plugin was designed with a modular architecture to be integrated with different SBSE tools. The plugin has standard interfaces to enhance its capabilities without requiring extensive modifications in the integrated tools. A usage example describes integrating with the tool OPLA-Tool for *Product Line Architecture (PLA)* search-based design. This integration aids non-specialist DMs when configuring objective functions in OPLA-Tool. This study demonstrates how the AIssistDM integrates SBSE tools with ChatGPT by detailing the system architecture and core functionalities. The AIssistDM represents an advancement in the SBSE field, offering a new way to support the DM's choice of parameters for SBSE tools.
Video Available on https://doi.org/10.6084/m9.figshare.25942801.v1

KEYWORDS

Search-based Software Engineering, Large Language Model, Plugin

1 INTRODUCTION

Search-Based Software Engineering (SBSE) employs metaheuristic optimization algorithms to solve complex problems in various domains of software engineering. SBSE applies methods that are particularly suited to address issues where traditional methods may fail, especially under conditions that demand optimization with conflicting constraints [13]. SBSE tools allow automatic support for SBSE solutions, addressing a great variety of tasks in different software engineering areas. Among these tools, we can mention EvoSuite, for automatic test data generation in different scenarios [11]; OPLA-Tool, for optimization of *Product Line Architecture (PLA)* design [12]; and OptiJIT, for optimizing just-in-time compilation strategies in order to enhance execution performance [16].

One significant challenge within SBSE, mainly for non-specialist *Decision-Makers (DMs)*,[1] is configuring optimal parameter values [1]. SBSE tools often deal with a broad spectrum of parameters, ranging from straightforward algorithmic settings to complex problem-domain configurations that significantly influence the outcomes of software engineering tasks [4]. For instance, algorithm-specific parameters of evolutionary algorithms include population size, mutation rate, and crossover probability, which directly affect the behavior and performance of the optimization process.

Problem-domain parameters in SBSE are specific to the software engineering context and impact the type and quality of the solutions generated. Those parameters often involve decisions that cannot be fully automated and require human expertise to balance trade-offs according to specific design needs and goals. The complexity of such parameters makes crucial to provide both automation and insightful support that guides non-specialist users through decision-making. One comprehensive example of such parameters is the objective functions used to evaluate the quality of solutions. More specific examples include Requirement Prioritization, which determines the criticality of software requirements during optimization; Architectural Constraints, which define allowable configurations of software architectures; Test Coverage Goals, specifying the extent of code coverage required for test generation; and Performance Benchmarks, setting specific performance targets to be met.

[1]The term "Decision-Makers" in this context refers to users responsible for making strategic decisions in the SBSE tool, including selecting appropriate parameters and evaluating potential solutions.

While some tools exist to assist in automatically configuring algorithmic parameters [17, 18], there remains a significant challenge in dealing with problem-domain parameters that require in-depth knowledge and strategic decision-making. This need for decision-making can pose a challenge for non-specialists who may not possess the knowledge required to interpret optimization results effectively. Guiding DMs through the decision-making process by simplifying data presentation and offering actionable insights could improve the application of SBSE tools [1]. Integrating advanced computational technologies, such as *Machine Learning (ML)*, offers promising opportunities to overcome these challenges. *Large Language Models (LLMs)*, a subset of ML, have demonstrated remarkable abilities in processing and understanding complex data sets [26], making them ideal for improving optimization tasks within SBSE tools. Their ability to generate context-aware solutions is particularly crucial in environments where traditional algorithms struggle to adapt dynamically to evolving conditions [24].

Considering these facts, this paper introduces AIssistDM, a plugin developed to enhance the optimization process implemented by a SBSE tool by integrating LLLM, more particularly ChatGPT[2]. This integration utilizes LLMs' natural language processing capabilities to offer intelligent, context-aware suggestions for parameters related to the domain problem in SBSE tools, such as objective functions and search operators, thus streamlining the search-based optimization process and improving decision-making [7].

Recognizing the challenges that non-specialist DMs face when configuring and utilizing sophisticated SBSE tools, the AIssistDM was specifically developed to provide accessible support. This plugin aims to be adopted in SBSE by simplifying the complexities involved in parameter configuration. This initiative is important as it ensures that the benefits of SBSE can be realized across a broader spectrum of DMs, regardless of their prior expertise.

AIssistDM was integrated as a case study with OPLA-Tool v2.0[12], for PLA optimization. The integration was made through a user-friendly interface that supports DMs in choosing objective functions. PLA design is a software engineering problem that significantly benefits from SBSE techniques due to its inherent complexity and the requirement to balance diverse and often conflicting design constraints [8]. Effective PLA design requires an approach to explore its complex trade-offs, making the application of SBSE techniques essential [20].

The AIssistDM Plugin aids DMs by leveraging ChatGPT to generate human-like text, simplifying the complex data associated with these decisions and enhancing the understanding and accessibility of parameter choices. ChatGPT can analyze and synthesize information from diverse data sources, providing tailored suggestions that respect each project's unique constraints and goals. This capability ensures that DMs are equipped with recommendations that are not only optimized but also contextually aligned with strategic business objectives [7].

By examining the system architecture and core functionalities, this study demonstrates how the AIssistDM Plugin can aid the DM in choosing problem-domain parameters of SBSE tools. Furthermore, the paper discusses how this integration advances the application of AI technologies within essential software engineering tasks, highlighting significant improvements in DM engagement in the optimization process.

The remainder of this paper is organized as follows: section 2 establishes background about LLMs used in this work. section 3 describes related work. section 4 presents the plugin's functionalities, structure, usage example and license. Finally, section 5 concludes the paper and outlines future research directions.

[2]https://chatgpt.com

2 BACKGROUND

Large Language Models (LLMs), such as the *Generative Pre-trained Transformer (GPT)* series developed by OpenAI[3], have significantly advanced natural language processing capabilities. Starting from GPT and evolving through GPT-2, GPT-3, and the latest iterations, these models have been trained on diverse internet text to generate coherent, contextually relevant text based on the prompts they receive [7, 21]. These models can perform tasks beyond simple text generation, including answering questions, summarizing lengthy documents, and generating code snippets, making them versatile AI research and application tools.

ChatGPT, a variant of these models, has been explicitly fine-tuned for dialogue applications, enabling more human-like interactions. Its underlying technology, based on deep learning architectures such as transformers, allows it to understand and generate responses by considering the broader context of a conversation or a query [24].

In SBSE, LLMs such as ChatGPT can be utilized to interpret complex software engineering tasks, automate the generation of coding artifacts, or even assist in developing and maintaining software by providing real-time recommendations and solutions. Their ability to process and synthesize large amounts of unstructured text data can enhance decision-making processes, bridging the gap between vast data handling and actionable software development insights [2].

3 RELATED WORK

Initial explorations into using LLMs in software engineering focused on tasks such as code completion [14], bug fixes and code refactoring [3], as well as automated documentation, where their ability to understand and generate human-like text proved particularly beneficial [7, 10]. These foundational applications demonstrated the versatility of LLMs in handling complex tasks that traditionally required extensive human expertise.

Zhang et al. [26] explore using LLMs for adaptive test case generation, showcasing how these models can dynamically generate tests to match evolving software requirements during development cycles. This application underscores the potential of LLMs to improve the responsiveness of testing processes to ongoing changes in software projects. On the other hand, Alon et al. [3] focus on the role of LLMs in automating routine maintenance tasks such as bug fixes and code refactoring. Their work highlights the capability of LLMs to understand and manipulate code, thereby facilitating the maintenance phase of software development with reduced human intervention.

These studies exemplify the adaptability of LLMs to diverse software engineering contexts, particularly their ability to automate

[3]https://openai.com

complex and repetitive tasks across various stages of the software development lifecycle. But unlike the general applications of LLMs discussed above, AIssistDM, proposed in this work, targets the optimization tools within SBSE. The plugin is designed to assist DMs, particularly non-specialists, in configuring optimal tool parameters, thereby easing the cognitive load on DMs. By leveraging LLMs like ChatGPT, the plugin interprets DM queries and provides context-aware recommendations, thus bridging the gap between complex optimization needs and user-friendly interfaces. We have not found an LLM-based tool with similar goals in the literature. As far as we know, our plugin is the first to assist DM in choosing SBES tools' parameters through integration with ChatGPT.

4 PROPOSED PLUGIN

This section presents the functionalities, structure and usage of AIssistDM. The source code, documentation, and instructions for building, installing, and usage can be obtained from Github [4].

4.1 Functionalities and Users

The AIssistDM Plugin is designed to assist DMs by integrating LLMs to suggest parameters for SBSE tools. This subsection outlines the main functionalities of the plugin, and identifies its potential users and applicability contexts.

4.1.1 Main Functionalities.

- **Integration with Existing Tools:** the AIssistDM Plugin is designed to be integrated with existing SBSE tools, such as: [6] [12] [15] [19] [22] [23]. This integration capability extends the utility of traditional SBSE tools by incorporating advanced AI-driven insights, thereby broadening the scope of their application. The plugin acts as a middleware that can interact with the tools and handle external API calls, making it a flexible addition to commercial and open-source software engineering environments. Section 4.3 presents an example of the protocol the tool must have to integrate within the plugin. It is important to detach that, for the integration to be possible, the SBSE tool must support external plugins or extensions.
- **Automated Parameters Suggestion:** The main functionality of AIssistDM is the suggestion of parameters for the integrated SBSE tool. Since the plugin acts as a bridge between the ChatGPT and the SBSE tool, the ChatGPT must be trained on the suggested parameters. Also, providing examples of parameter usage to ChatGPT is important so the model can correctly make suggestions. subsection 4.3 presents an example of using the AIssistDM plugin to suggest objective functions in a SBSE tool.
- **Interactive Suggestions:** DMs can interactively visualize the parameter suggestions through a graphical interface in the SBSE tool. The interface helps the DM better understand the trade-offs between different parameter choices and make informed decisions.

4.1.2 Potential Users. This work adopts the term DM as standard since the tool's user in the context of optimization in SBSE is commonly referred to as DM. The plugin was designed to assist DMs, mainly non-specialist DMs. These DMs can be Software Engineers involved in creating solutions and using the plugin to enhance decision-making, and even Research Scholars engaged in studying and developing SBSE approaches and tools who can use the plugin to experiment with different optimization techniques.

4.2 Overall Architecture

This subsection details the architecture and core components of the AIssistDM Plugin, describing how these elements work together. The plugin is designed to integrate language model capabilities into existing SBSE tools, prioritizing modularity, scalability and maintainability, as presented in Figure 1. We can see in this figure that the Optimization Tool, the ChatGPT, and the AIssistDM Plugin interact through interfaces. The focus of this work is the plugin, which can be expanded into the following components:

- **Popup** controls the plugin's opening and the buttons' actions of connecting to the WebSocket and sending test messages to the ChatGPT.
- **Background** controls the WebSocket connection, responsible for exchanging messages between the optimization tool and the ChatGPT.
- **WebSocket Server** is a server application that enables two-way, persistent communication channels over a single TCP (Transmission Control Protocol) connection. In this work, we used the WebSocket for real-time communication between the optimization tool in the server and the ChatGPT in the web browser.

As mentioned and presented in Figure 1, a WebSocket integrates the optimization tool with the AIssistDM Plugin to exchange questions and answers between the optimization tool and the ChatGPT. The *WebSocketConnection* class, included in the *Background* component, has the socket instance and the methods of controlling the socket, such as opening, closing, exchanging messages, and treating errors.

To send messages and get ChatGPT answers, two functions in *Background* were developed. The *Send Message Function* (Figure 1) presents the flow of sending messages to the ChatGPT. The first step is inspecting the input field in the ChatGPT application responsible for exchanging messages. Figure 2 presents an example of a visual inspection of this input field. As can be seen, this field has an HTML ID that can be obtained through Javascript coding. After obtaining the input field, the second step is to put the received message in the field and dispatch an event using the button to send questions.

Getting messages from the ChatGPT is more advanced since the answers are asynchronous, which demands a timer that stays checking if the answer has arrived. This is made by the *Wait Answer Function* presented in Figure 1. For the ChatGPT, we tested different configurations for the timer, and we verified that the best option was to check every 3 seconds to see if the answer arrived. Practically, every time ChatGPT answers questions, as presented in Figure 3, it creates a button to stop generating questions. So, to know if the ChatGPT is answering yet, the plugin needs to check if this button is on the page.

In summary, the AIssistDM plugin provides a graphical interface that allows DMs to interact, configure parameters, and visualize optimization results. At the plugin's core, this engine integrates the

[4]https://github.com/otimizes/AIssistDM-plugin

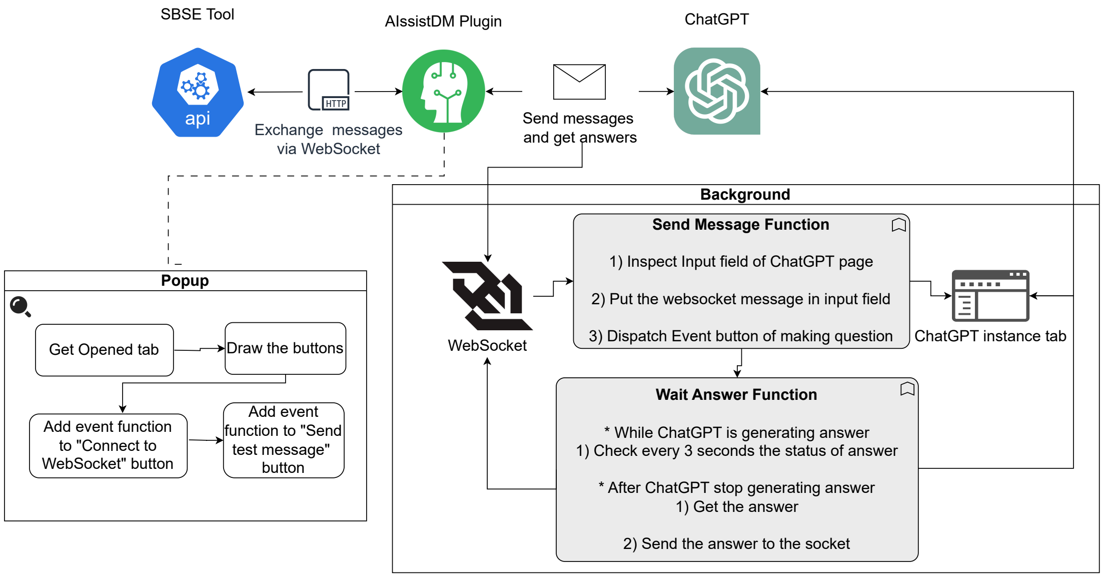
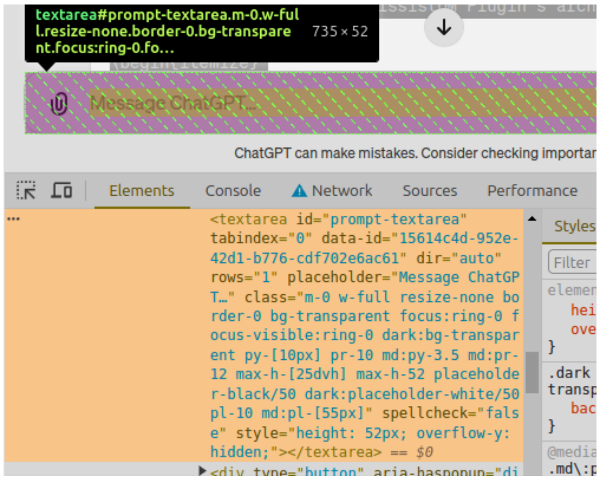
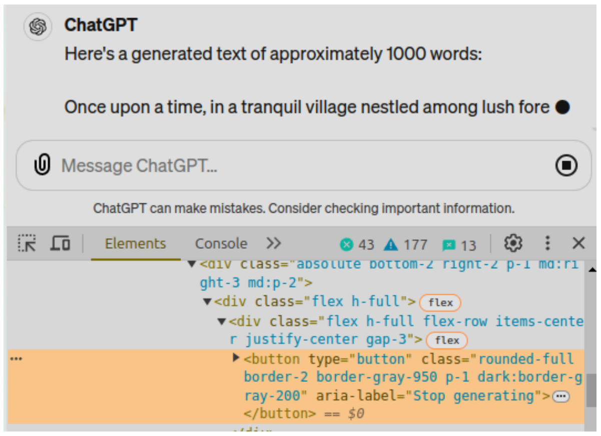

SBSE Tool
api
HTTP
Exchange messages via WebSocket
AIssistDM Plugin
Send messages and get answers
ChatGPT
Background
Send Message Function
1) Inspect Input field of ChatGPT page
2) Put the websocket message in input field
3) Dispatch Event button of making question
WebSocket
ChatGPT instance tab
Wait Answer Function
* While ChatGPT is generating answer
1) Check every 3 seconds the status of answer
* After ChatGPT stop generating answer
1) Get the answer
2) Send the answer to the socket
Popup
Get Opened tab
Draw the buttons
Add event function to "Connect to WebSocket" button
Add event function to "Send test message" button

Figure 1: High-level architecture diagram of the AIssistDM Plugin


textarea#prompt-textarea.m-0.w-full.resize-none.border-0.bg-transparent.focus:ring-0.fo...
735 × 52
Message ChatGPT...
ChatGPT can make mistakes. Consider checking importar
Elements
Console
Network
Sources
Performance
Styles
Filter

```
<textarea id="prompt-textarea"
tabindex="0" data-id="15614c4d-952e-
42d1-b776-cdf702e6ac61" dir="auto"
rows="1" placeholder="Message ChatGP
T…" class="m-0 w-full resize-none bo
rder-0 bg-transparent focus:ring-0 f
ocus-visible:ring-0 dark:bg-transpar
ent py-[10px] pr-10 md:py-3.5 md:pr-
12 max-h-[25dvh] max-h-52 placeholde
r-black/50 dark:placeholder-white/50
pl-10 md:pl-[55px]" spellcheck="fals
e" style="height: 52px; overflow-y:
hidden;"></textarea> == $0
```

Figure 2: Inspecting ChatGPT input field.



Here's a generated text of approximately 1000 words:

Once upon a time, in a tranquil village nestled among lush fore

ChatGPT can make mistakes. Consider checking important information.

```
▼<div class="absolute bottom-2 right-2 p-1 md:ri
ght-3 md:p-2">
  ▼<div class="flex h-full">
    ▼<div class="flex h-full flex-row items-cente
    r justify-center gap-3">
      ▶<button type="button" class="rounded-full
      border-2 border-gray-950 p-1 dark:border-g
      ray-200" aria-label="Stop generating">
      </button> == $0
```

Figure 3: Inspecting ChatGPT answering questions

capabilities of ChatGPT, managing the processing and generation of suggestions based on DM inputs and project data. The plugin handles all data interactions between the optimization tool and ChatGPT, ensuring data integrity and security.

The plugin is designed to integrate into SBSE tools that support external plugins or extensions by using Websocket. Integration typically requires minimal changes to the host application, as the AIssistDM Plugin adheres to standard data exchange formats and communication protocols. Section 4.3 presents the usage example of the plugin and how it was integrated into OPLA-Tool v2.0.

4.3 Usage

The AIssistDM Plugin provides an intuitive and effective interface for enhancing SBSE processes with AI-driven capabilities. This subsection offers a detailed usage example, demonstrating how DMs can leverage the optimization tool using SBSE techniques augmented by LLMs.

4.3.1 Search-Based PLA design. As this paper presents a plugin that can be integrated into SBSE tools, we choose a search-based PLA design tool (OPLA-Tool v2.0 [12]) to demonstrate the AIssistAI usage. All the integration made with OPLA-Tool v2.0 is available on Github [5]. Search-based PLA design represents a specific application area where SBSE techniques have shown significant potential [9].

[5]https://github.com/otimizes/OPLA-Tool

The complexities inherent in PLA design, such as feature modularization, reuse, variability, and extensibility, present challenges well-suited to search-based approaches' capabilities.

Adopting SBSE in PLA design is necessary due to the intrinsic complexities of managing diverse architectural elements and their dependencies. SBSE's capability to explore extensive design spaces through optimization algorithms enables architects to discover innovative solutions that traditional heuristic-based methods might miss. This exploratory capacity is critical in PLA design, where the optimal integration of features can drastically impact the final product line's performance and maintainability.

OPLA-Tool v2.0 tool is implemented in the Java programming language with Spring framework[6]. This version employs various search algorithms to optimize PLAs by balancing conflicting objectives such as Class Coupling (ACLASS), Feature Modularization (FM), and Cohesion (COE). A key component of OPLA-Tool is its evaluation model, which integrates objective functions derived from software metrics to measure aspects such as modularity, extensibility, and variability [25]. Despite the efficacy of OPLA-Tool v2.0, there is still a gap in efficiently selecting and tuning the myriad of possible objective functions to optimize software engineering outcomes [12].

4.3.2 Installation. ChatGPT works on a web browser. In this sense, DMs must install the AIssistDM Plugin in their browser navigator to enable communication between the ChatGPT and the SBSE tool. The repository on GitHub has a tutorial on how the plugin can be installed. Figure 4 presents an example of manual installation for Google Chrome [7]. To install, the DM must follow the steps: (1) Access the URL of extensions (chrome://extensions/); (2) Click on the "Load unpacked button"; and (3) select the plugin folder;

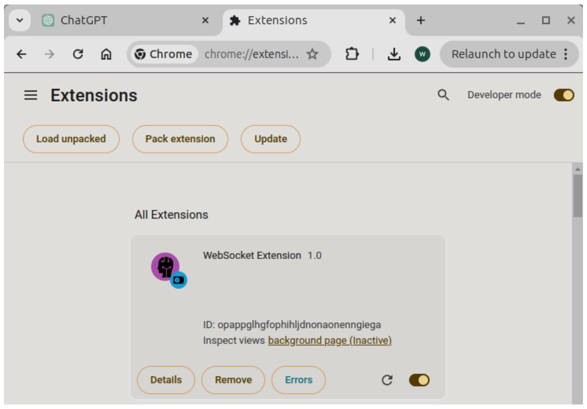


Figure 4: Manual installation of the AIssistDM Plugin in Google Chrome.

Once integrated, the plugin is accessible from the browser, where it can be activated by selecting the AIssistDM Plugin. As presented in Figure 5, the DM must access ChatGPT, log in, and click on the extension.

[6]The Spring Framework is an application framework and inversion of control container for the Java platform, commonly used to develop web applications.
[7]https://www.google.pt/intl/en-US/chrome

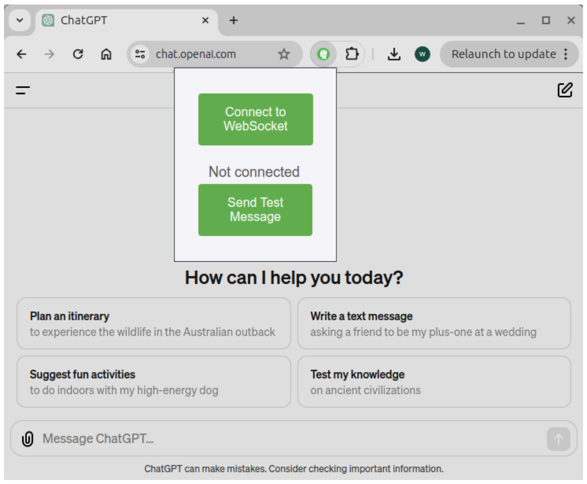

ChatGPT
chat.openai.com
Relaunch to update
Connect to WebSocket
Not connected
Send Test Message
How can I help you today?
Plan an itinerary
to experience the wildlife in the Australian outback
Write a text message
asking a friend to be my plus-one at a wedding
Suggest fun activities
to do indoors with my high-energy dog
Test my knowledge
on ancient civilizations
Message ChatGPT...
ChatGPT can make mistakes. Consider checking important information.

Figure 5: Main GUI (Graphical User Interface).

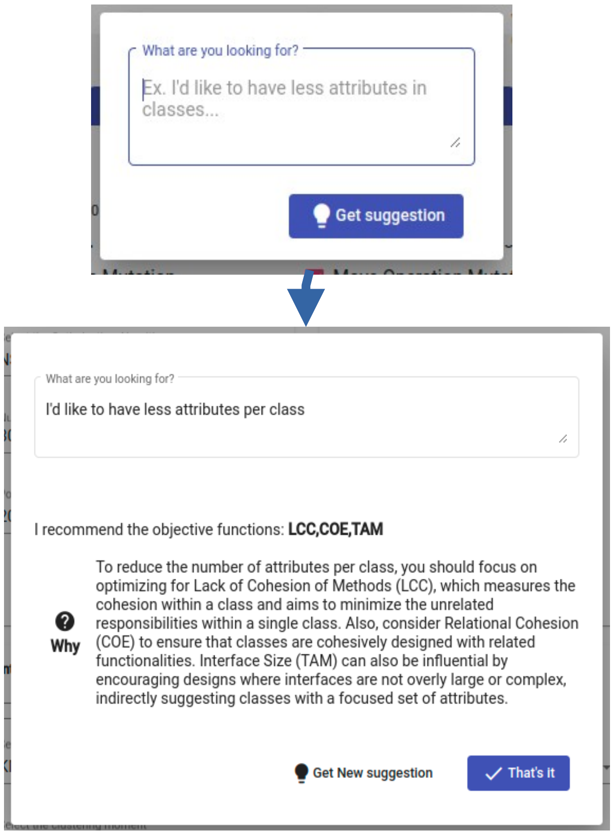
The DM has two options at the plugin's main GUI: The first one is to make a Websocket server available on port 3000, and any optimization application can connect to this port to exchange JSON messages. The second option is to send a test message to ChatGPT.

4.3.3 OPLA-Tool Adaptation. Figure 6 presents an example of the integration with OPLA-Tool. The plugin's core is a dynamic interaction model between the DM and the LLM. The process initiates with the DM requesting suggestions for objective functions when he/she clicks the "Get a suggestion" button. This request is translated into a structured prompt and dispatched to ChatGPT by the plugin, which processes the query and returns a list of suggested objective functions. All the questions are accompanied by an instruction to receive the answer in a specified JSON format: `{"fns": ["..."], "suggestion": "..."}`. Here, `fns` denotes the objective functions, represented as uppercase acronyms, and `suggestion` provides detailed recommendations. This format was chosen due to its simplicity and ease of integration into existing systems, ensuring that suggestions are accessible and actionable via API (Application Programming interface).

4.3.4 Training ChatGPT for assisting DMs . All the potential of the AIssistDM plugin lies in the data that the DM of the optimization tool will supply to the ChatGPT in the context in which the plugin will be used. In this sense, for using the AIssistDM Plugin, it is necessary to tailor the underlying LLM, specifically ChatGPT, to understand better and respond to the specific needs of software engineering tasks. Training ChatGPT involves fine-tuning it with datasets relevant to the functions it will perform, including diverse software engineering problems and optimization scenarios.

The training process typically involves several steps:

(1) **Data Collection:** Gathering a comprehensive dataset that includes various software engineering texts, such as code snippets, documentation, and problem descriptions.
(2) **Preprocessing:** Cleaning and preparing the data to ensure it is suitable for training, which may involve removing irrelevant information, correcting errors, and formatting the data in a digestible way for the model.


What are you looking for?
Ex. I'd like to have less attributes in classes...
Get suggestion
What are you looking for?
Get New suggestion
That's it
Why

I'd like to have less attributes per class

I recommend the objective functions: **LCC,COE,TAM**

To reduce the number of attributes per class, you should focus on optimizing for Lack of Cohesion of Methods (LCC), which measures the cohesion within a class and aims to minimize the unrelated responsibilities within a single class. Also, consider Relational Cohesion (COE) to ensure that classes are cohesively designed with related functionalities. Interface Size (TAM) can also be influential by encouraging designs where interfaces are not overly large or complex, indirectly suggesting classes with a focused set of attributes.

Figure 6: Running example of objective function suggestion.

(3) **Model Training:** Feeding the prepared dataset to ChatGPT allows it to learn from the specific patterns, terminology, and contexts used in software engineering.
(4) **Evaluation:** Testing the trained model on new, unseen data to assess its understanding and ability to generate accurate and relevant responses.
(5) **Iteration:** Based on performance, additional adjustments and refinements may be made to improve the model's accuracy and responsiveness.

By customizing ChatGPT, the AIssistDM Plugin becomes more proficient in handling the challenges of complex software engineering tasks. It improves in providing contextually appropriate suggestions that can significantly enhance decision-making processes within SBSE tools.

Training LLMs like ChatGPT for specific domains is essential to leverage their full potential and ensure they perform optimally in targeted applications. This adaptability underscores LLMs' versatility and capacity to transform industries by providing tailored, intelligent solutions.

4.3.5 Initial Evaluation with PLA Specialists. The AIssistDM was initially evaluated with two PLA specialists to assess the plugin's integration in the OPLA-Tool [5]. We verified the suggestions for objective functions made through the plugin. The first DM is a doctoral-level educator with advanced PLA design knowledge. The second DM is a computer science graduate working in the industry with moderate knowledge of relevant fields. This preliminary test aimed to gather DM feedback on the practical utility and performance of the plugin in a real-world PLA optimization scenario.

During this evaluation, the DMs interacted with the AIssistDM, engaging with its features and assessing the relevance and accuracy of the suggestions provided. The feedback was highly encouraging, with the DMs highlighting several key observations. About usage, DM 1 said, *"The interface is user-friendly and intuitive"*. Regarding the quality of suggestions, DM 2 said, *"I was surprised. The module brought answers that, from my point of view, surpass the suggestion of a trained DM with no experience analyzing optimized solutions."*.

These comments indicate that the AIssistDM Plugin can deliver contextually appropriate suggestions that enhance the decision-making process in PLA design. This promising start lays a solid foundation for further development and more comprehensive evaluations to refine the plugin's capabilities and ensure its efficacy in diverse SBSE tools.

The AIssistDM plugin provides advantages over directly using ChatGPT, including automated background training phases, seamless integration with SBSE tools, and tailored context-aware parameter suggestions. Unlike a standalone ChatGPT session, the plugin offers a user-friendly interface and ensures consistency in parameter selection by maintaining a history of interactions and learning from previous decisions. This integrated approach enhances decision-making and reduces the cognitive load on DMs.

4.4 License

AIssistDM is an open-source plugin under the AGPLv3 license (GNU Affero General Public License v3.0).

5 CONCLUDING REMARKS

This work presented a plugin designed to integrate LLM into SBSE tools. The AIssistDM Plugin showcases how AI can improve optimization approaches, assisting the DM in the decision-making of the SBSE tool's parameters. The plugin implementation has shown that including it significantly enhances decision-making by providing intelligent, context-aware suggestions for tool parameters. This has streamlined the optimization tasks and improved the tools' usability, enabling novice and experienced DMs to achieve better optimization results with less effort.

Future work could explore: (i) integrating other SBSE tools to address different areas of SBSE; (ii) improving of the AI models used in the plugin, including training on larger datasets and refining the algorithms to understand complex software engineering contexts better; (iii) investigating the use of the plugin in other SBSE tools, which explore different software engineering domains; and (iv) conducting comprehensive DM studies to evaluate the impact of the AIssistDM Plugin on the workflow of DMs, aiming to optimize the interface and functionalities based on DM feedback.

ACKNOWLEDGMENTS

This research is supported by CAPES (Grant: 88887.941765/2024-00) and CNPq (Grant: 310034/2022-1, 404027/2023-7).

REFERENCES

[1] Wasif Afzal, Richard Torkar, and Robert Feldt. 2009. A systematic review of search-based testing for non-functional system properties. *Information and Software Technology* 51, 6 (2009), 957–976.

[2] Baleegh Ahmad, Shailja Thakur, Benjamin Tan, Ramesh Karri, and Hammond Pearce. 2024. On Hardware Security Bug Code Fixes By Prompting Large Language Models. *IEEE Transactions on Information Forensics and Security* (2024).

[3] Uri Alon, Roy Sadaka, Omer Levy, and Eran Yahav. 2020. Structural language models of code. In *International conference on machine learning*. PMLR, 245–256.

[4] Andrea Arcuri and Gordon Fraser. 2013. Parameter tuning or default values? An empirical investigation in search-based software engineering. *Empirical Software Engineering* 18 (2013), 594–623.

[5] Anonymous Authors. 2024. Complementary Material. (2024). https://figshare.com/s/3f89f0bce25aaa406cdc

[6] Cosimo Birtolo, Paolo Pagano, and Luigi Troiano. 2009. Evolving colors in user interfaces by interactive genetic algorithm. In *2009 World Congress on Nature & Biologically Inspired Computing (NaBIC)*. IEEE, 349–355.

[7] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems* 33 (2020), 1877–1901.

[8] Paul Clements and Linda Northrop. 2002. *Software Product Lines: Practices and Patterns*. Addison-Wesley Professional.

[9] Thelma Elita Colanzi, Silvia Regina Vergilio, Itana Gimenes, and Willian Nalepa Oizumi. 2014. A search-based approach for software product line design. In *Proc. of the 18th International Software Product Line Conference-Volume 1*. 237–241.

[10] Jacob Devlin et al. 2018. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *North American Chapter of the Association for Computational Linguistics* (2018).

[11] Gordon Fraser and Andrea Arcuri. 2011. EvoSuite: Automatic Test Suite Generation for Object-Oriented Software. In *Proceedings of the 19th ACM SIGSOFT Symposium and the 13th European Conference on Foundations of Software Engineering*. ACM, 416–419.

[12] Willian Marques Freire, Mamoru Massago, Arthur Cattaneo Zavadski, Aline Maria Malachini, Miotto Amaral, and Thelma Elita Colanzi. 2020. OPLA-Tool v2. 0: a tool for product line architecture design optimization. In *Proceedings of the XXXIV Brazilian Symposium on Software Engineering*. 818–823.

[13] Mark Harman and Bryan F Jones. 2010. Search based software engineering: Trends, techniques and applications. *ACM Computing Surveys (CSUR)* 45, 1 (2010), 11.

[14] Maliheh Izadi, Jonathan Katzy, Tim Van Dam, Marc Otten, Razvan Mihai Popescu, and Arie Van Deursen. 2024. Language Models for Code Completion: A Practical Evaluation. In *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering*. 1–13.

[15] Wael Kessentini, Manuel Wimmer, and Houari Sahraoui. 2018. Integrating the designer in-the-loop for metamodel/model co-evolution via interactive computational search. In *Proceedings of the 21th ACM/IEEE International Conference on Model Driven Engineering Languages and Systems*. 101–111.

[16] William B. Langdon and Justyna Petke. 2017. Optimising Existing Software with Genetic Programming. *IEEE Transactions on Evolutionary Computation* 21, 1 (2017), 118–135.

[17] William B Langdon and Justyna Petke. 2018. Evolving better software parameters. In *Search-Based Software Engineering: 10th International Symposium, SSBSE 2018, Montpellier, France, September 8-9, 2018, Proceedings 10*. Springer, 363–369.

[18] Manuel López-Ibáñez, Jérémie Dubois-Lacoste, Thomas Stützle, and Mauro Birattari. 2011. *The IRACE package, Iterated Race for Automatic Algorithm Configuration*. Technical Report TR/IRIDIA/2011-004. IRIDIA, Université Libre de Bruxelles, Belgium.

[19] Mohamed W Mkaouer, Marouane Kessentini, Slim Bechikh, and Daniel R Tauritz. 2013. Preference-based multi-objective software modelling. In *2013 1st International Workshop on Combining Modelling and Search-Based Software Engineering (CMSBSE)*. IEEE, 61–66.

[20] Klaus Pohl, Günter Böckle, and Frank van der Linden. 2005. *Software Product Line Engineering: Foundations, Principles, and Techniques*. Springer Science & Business Media.

[21] Alec Radford and Karthik Narasimhan. 2018. Improving Language Understanding by Generative Pre-Training. https://api.semanticscholar.org/CorpusID:49313245

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

[23] Paolo Tonella, Angelo Susi, and Francis Palma. 2013. Interactive requirements prioritization using a genetic algorithm. *Information and software technology* 55, 1 (2013), 173–187.

[24] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*. 5998–6008.

[25] Yenisei D. Verdecia, Thelma E. Colanzi, Silvia R. Vergilio, and Marcelo C.B. dos Santos. 2017. An Enhanced Evaluation Model for Search-based Product Line Architecture Design.. In *CIbSE*. 155–168.

[26] Chen Yang, Junjie Chen, Bin Lin, Jianyi Zhou, and Ziqi Wang. 2024. Enhancing LLM-based Test Generation for Hard-to-Cover Branches via Program Analysis. *arXiv preprint arXiv:2404.04966* (2024).