

**TEXAS A&M INTERNATIONAL UNIVERSITY**

**School of Engineering**

**ChatGPT Paper**

**Software Engineering**

**CSCE 4201**

**Instructor: Dr. Mustafa Al Lail**

**Alexis Huante, Vilen C. Elliott**

**Abstract**

As of today's date, Chatbots are being integrated into several application systems, such as banking, marketing, and healthcare. Instead of relying on people to answer questions, Chatbots can provide efficient responses to these questions. In this project, we incorporated ChatGPT into the Temporal Property Validator (TPV), a tool designed to validate UML class diagrams. Our implementation enables users to obtain general assistance from the chatbot, such as guidance on creating UML class diagrams. For the implementation, we used GitHub repositories as a reference, where ChatGPT understands and answers a question in Java, and we created a plugin in the UML-based specification environment (USE) to implement the plugin into the TPV tool. Once the JAR plugin has been created, we transfer it to the TPV's plugins folder, enabling ChatGPT to run within the tool.

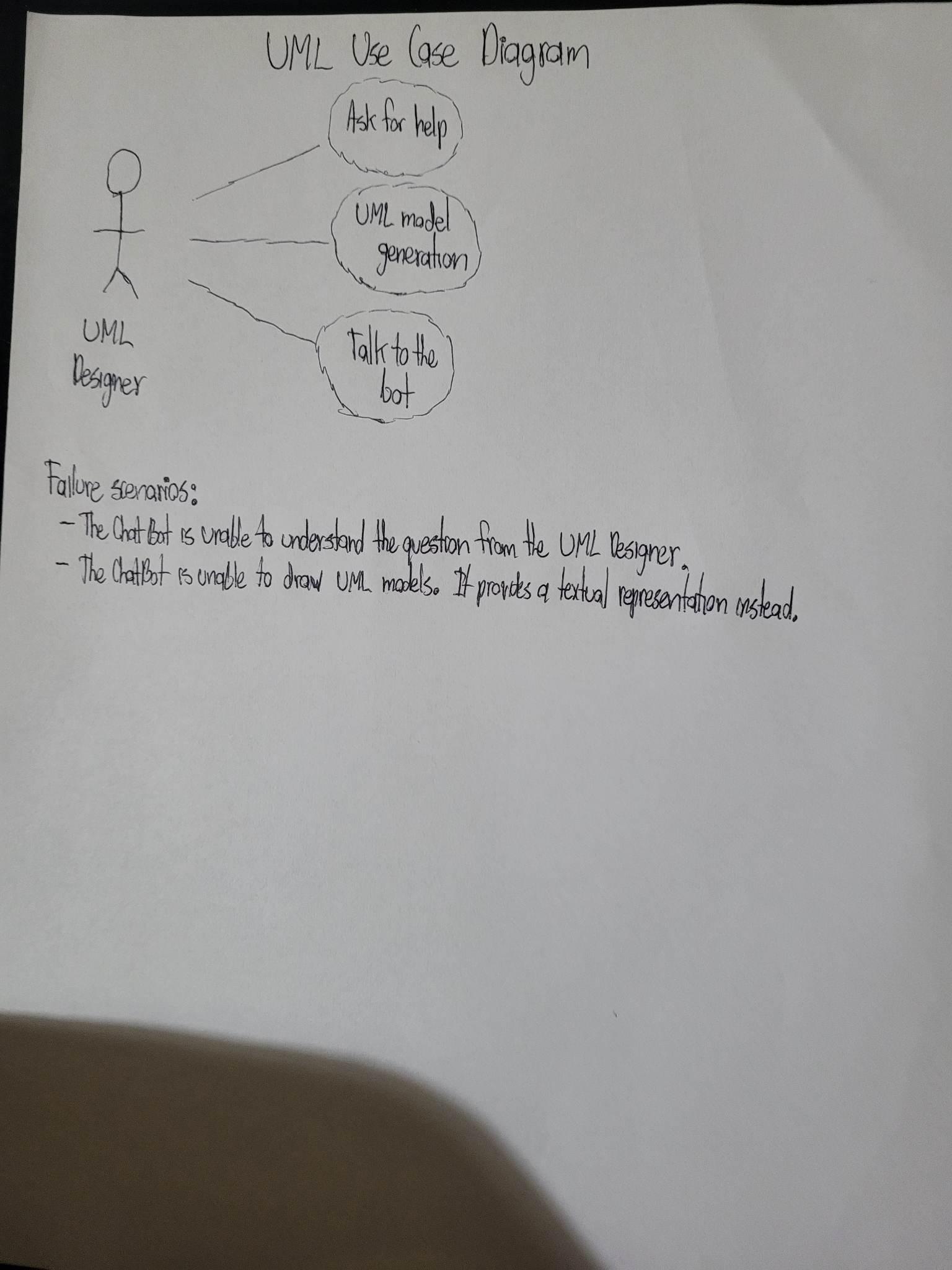
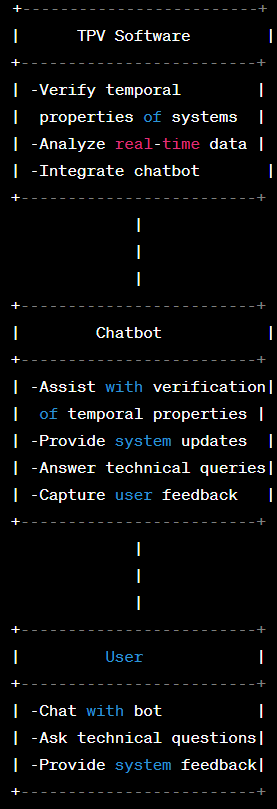
1. **Introduction**

In recent years, chatbots have become an increasingly popular tool for businesses to enhance their customer service experience. With the rise of artificial intelligence and natural language processing, chatbots have evolved to become more sophisticated and intelligent. Among the most advanced chatbots in the market today is ChatGPT, a language model powered by the GPT-3.5 model developed by OpenAI.

This report details the implementation of a ChatGPT-powered chatbot into the USE 6.0 TPV software, designed to provide a more seamless and personalized user experience. With the integration of ChatGPT, the TPV software can now offer users a more human-like interaction, enabling them to converse with the chatbot in natural language and receive relevant and helpful responses to their queries.

The following sections will provide an overview of the use cases for the chatbot, the methodology used during development, the implementation process into the TPV tool, and the results of the integration. Additionally, we will also discuss the challenges faced during the implementation process and provide recommendations for future improvements.

1. **Requirements and use cases**



1. **Methodology**

In this section, we are going to talk about how ChatGPT was created in Java and how it was implemented into the TPV tool to help people.

**3.1. ChatGPT in Java**

First of all, we used code from the GitHub repository by Antoine Gauthier as a start guide. However, with that code, ChatGPT was not able to respond because we had to download a JSON file and include it as a dependency on our project. Also, we had to get an OpenAI key by creating an account on the OpenAI website that would charge us cents whenever we were asking a question to ChatGPT. This means that the program makes an HTTP request to the OpenAI API to receive a response, which requires an internet connection. With this, ChatGPT was finally able to respond to our questions in the console. But, instead of typing the question in the code and giving responses in the console, we decided to make a Graphical User Interface (GUI) as demonstrated in Figure 1, where the user can ask a question and ChatGPT can answer the question in the GUI. The GUI simply has a text box where the user can ask the question and a submit button, where it will make an HTTP request to the API and give a response.

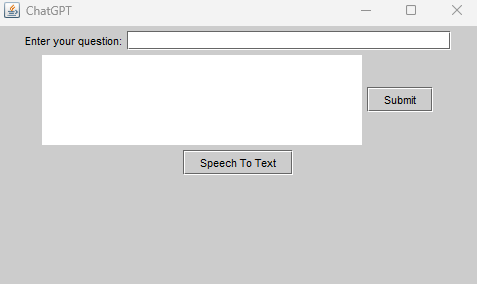


Figure 1. ChatGPT GUI

**3.2. ChatGPT Implementation in TPV tool**

To implement ChatGPT into the TPV tool, we needed to create a plugin in USE (UML-based Specification Environment). We used plugin examples from a GitHub repository by Andreas Kastner and Lars Hamann. Using their files as a guide, we were able to create the plugin. The following files needed to be changed: build.xml, useplugin.xml, manifest.mf, and ChatGPTPlugin.java. Once the files have been modified, we can then run the build.xml using the ant command by installing Apache Ant in the system. Finally, we know if we successfully created the plugin if we move it to the plugins folder in the TPV tool, run the tool, and see our plugin in the Plugins tab as shown in Figure 2.

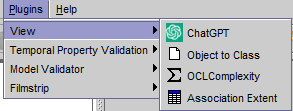


Figure 2. Plugins Tab of USE

1. **Progress and Challenges**

We have made significant progress towards implementing a chatbot into the TPV software. We created an OpenAI account and API key, which allowed us to use the OpenAI GPT-3.5 language model to power the chatbot. We also ported over a working prototype of the program from Python into Java, using our existing code and migrating it to Java. Additionally, we created a user interface using a built-in Java module, providing an intuitive interface for users to interact with the chatbot. In the end, we implemented the ChatGPT code as a plugin for the UML-based specification environment.

While making progress on the implementation of the chatbot into the TPV software, we encountered several challenges. We needed to learn new aspects of Java to complete the project, such as structuring the project in a way that was compatible with the TPV software, and creating a user interface using Java Swing. Additionally, we had to learn how to export the code as a JAR and add it into the TPV plugin folder, which involved ensuring that all the necessary dependencies were included in the JAR file. Despite these challenges, we were able to create the plugin and include the chatbot into the TPV software.

1. **Conclusion**

To conclude, the main goal, which is implementing the ChatGPT into the TPV tool was achieved. The most toughest task we encountered was the creation of the plugin, which involved modification of several files and coding. We collected code examples from GitHub repositories as a guide and managed to run ChatGPT in Java, create a GUI, and create a plugin for USE. Users who would like to use the TPV tool should now be able to receive answers without asking a human.

1. **Future Work**

While we have made significant progress towards implementing a chatbot powered by ChatGPT into the TPV software, there is still a lot of work to be done. Here are some of the key areas we plan to focus on in future iterations of the project:

Creating a better user interface: While we were able to create a simple user interface for the chatbot, there is clearly room for improvement. In future iterations of the project, we plan to invest more time and resources into creating a more clean and user-friendly interface that will make it easier for users to interact with the chatbot.

Adding Text-to-Speech & Speech-to-Text support: Currently, the chatbot only supports text-based interactions. In order to improve the user experience and make the chatbot more accessible, we plan to add support for Text-to-Speech and Speech-to-Text capabilities. This will allow users to interact with the chatbot using their voice, which can be particularly useful in scenarios where typing is difficult or impractical.

Teaching ChatGPT about TPV software: While ChatGPT is a powerful language model, it currently lacks knowledge about the TPV software. In future iterations of the project, we plan to invest more time into teaching ChatGPT about TPV software. This will involve training the language model on relevant data and integrating specific knowledge into its responses.

Overall, we believe that these future improvements will go a long way in making the chatbot a more useful and accessible tool for TPV software users. We would like to continue to work on the project and make improvements in the future.

**References**

<https://github.com/useocl/use_plugins> by Andreas Kastner and Lars Hamann

<https://gist.github.com/gantoin/190684c344bb70e5c5f9f2339c7be6ed> by Antoine Gauthier