Software Engineering Assignment

Ranim Khojah

0.1 Ranim's Research

With the increasing growth of software products and organizations, the activities associated with software engineering have also increased. These activities include tasks such as environment setup, writing helper code, team synchronization, and monitoring, among others. To enable software engineers to focus on the core process and development of the product, it becomes crucial to automate these activities. In addition to automation, software engineers require assistance in adapting to evolving software changes and incorporating new technologies.

To provide such assistance, software bots for software development, known as DevBots, can be employed. DevBots are automation tools that exhibit human-like traits, including language capabilities, autonomy, and intelligence. When DevBots interact with humans using natural language, they are classified as chat-based DevBots, or Chatbots for short.

However, while Chatbots theoretically have the potential to assist software engineers, they face several challenges that hinder their effectiveness and efficiency. One main limitation of Chatbots is that they often suffer from limited vocabulary and domain understanding, making them not much more useful than conducting a Google search or typing some commands into a command line. Consequently, users may question the reliability and trustworthiness of Chatbots.

Moreover, software engineering involves various activities specific to different roles and stages in the software development lifecycle. These activities may fall under the same categories but require different outcome and help from the Chatbot depending on the role. For example, consider the activities related to bug fixing and testing. While both activities fall under the category of quality assurance, software engineers may expect different kind of assistance. n the case of a software tester, the Chatbot can assist by providing automated test case generation, test execution, and result analysis. While, for a software developer responsible for bug fixing, the Chatbot can provide intelligent code analysis and debugging support.

So, havnig one Chatbot that can assist in Software engineering in general with all its different roles and stages, may be challenging when it comes to incorporating relevant context to get the intended help.

Considering these challenges, my research project aims to gain a deeper understanding of the issues, examine them in detail, and propose potential solutions. Specifically, we will conduct empirical studies to determine the needs of software engineers, identify the vocabulary developers require, and develop a framework to train this vocabulary based on project context and domain. In addition, we will consider crucial human factors, such as trust and satisfaction, when designing and utilizing the Chatbot.

Furthermore, we will explore how Chatbots can be integrated more effectively into existing state-of-the-art software development workflows, such as Agile and DevOps. Our goal is to propose a stable and comprehensive framework for the design, development, and evaluation of Chatbots in the software engineering domain.

0.2 From Robert's lectures

0.2.1 AI/ML process/workflow

Machine learning (ML) applications go through nine stages according to Amershi et al.

Chatbots are one example of ML applications that follow the same process. Some differences are present in the type of data collected (mostly text labelled with intents) and the way we process the data (using NLP techniques) and the model we train is usually an NLU (natural language understanding) model that performs intent classification on the input from the user. Going back to the challenges mentioned in section 0.1, we may need chatbots that are context-aware and multi-modal to cover different areas in Software engineering and provide more effective responses. When creating multimodal Chatbots, we will focus mostly on the Data Collection and Model Training stages. In the Data collection we would consider gathering data that includes text, code, diagrams ...etc. Consequently, we would need to develop ML models that can handle and learn meaningful patterns from multiple modalities.

Regarding context-aware Chatbots, more work would be done during the Feature Engineering stage. Where the focus would be on extracting relevant contextual features from conversation history, the development environment to enhance the chatbot's understanding and usefulness of the response. One example of context-aware Chatbots that were developed shortly before the release of ChatGPT is The Programmer's Assistant [1]. The Programmer's Assistant considers the conversation history and the code in the IDE as context to make the responses more relevant to the developers. The authors did not perform Feature Engineering as they used a pre-existing model i.e., OpenAI's Codex rather than building their own and designing the features and parameters that the model uses.

Finally, after building such chatbots, we can develop metrics in the Model Evaluation stage to assess bias and overfitting in chatbot responses.

0.2.2 ML invariant/metamorphic testing

ML invariant/metamorphic testing can be applied to chatbots as a part of an evaluation to ensure consistent behavior in different scenarios. For chatbots, metamorphic relations can be defined based on the expected behavior from the chatbot. For example, if a chatbot is expected to retrieve information, then a metamorphic relation could be that the information provided by the chatbot shall remain consistent even if the user's prompt is mutated, e.g., rephrased using different terminologies and synonyms.

ML invariant/metamorphic testing helps identify potential issues, such as sensitivity to slight changes in input or inconsistencies in the chatbot's responses, enabling developers to improve the robustness and reliability of the chatbot, more specifically, its NLU model.

0.3 From Linda's guest lecture

0.3.1 Bots and Productivity in Software Engineering

The guest lecturer talked about two main concepts, that is, the definition of a Bot and the productivity of software engineers. There are many tools and technologies that are being called ”Bots” because of certain characteristics. For example, some tools on GitHub are called ”Bots” since they automatically sync projects between different platforms e.g., Syncing overleaf projects on GitHub. Therefore, since the term ”Bot” has been used without a specific basis, the guest lecturer presents three main personas that define a Bot in their research. A bot can be called so if it at least one of these human traits: 1- Smartness: If a bot has the sense of intelligence (Often by using AI) that it uses to make decisions and perform tasks, then the bot fits the persona ”Sam” 2- Autonomy: If the bot automatically performs tasks given a trigger, the bot fits the persona ”Alex” 3- Chat-ability: If the bot has the language aspect incorporated in the interaction with the human, then the bot fits the persona ”Charlie”

A Chatbot is mainly ”Charlie” as it mainly uses natural language to communicate with humans. However, these personas are not mutually exclusive. A Chatbot can still have other elements of autonomy and intelligence that are involved when performing tasks. One Software Engineering idea that could be applied to their personas is the establishment of certain practices when it comes to the process followed when developing the Chatbot or the automated testing of chatbots based on the persona. For example, testing ”Alex” would focus on testing triggers while testing ”Sam” would focus on consistencies.

The guest lecturer also talked about how to measure the productivity of software developers and pointed out that it is not easy to quantify in one measure. Therefore, the SPACE framework can be used. SPACE framework can help better assess the productivity of developers using different dimensions: satisfaction and well-being; performance; activity; communication and collaboration; and efficiency and flow. Chatbots that are used in Software development can be assessed using this framework to see if the chatbots are affecting the developer’s productivity in any way.

0.4 From The List

0.4.1 Security and Privacy

Security and privacy are gaining tremendous focus in the software engineering field recently. This focus is driven by the growing awareness of the potential risks and challenges associated with digital systems and the need to ensure the protection of user data. Security (or cybersecurity) concerns taking measures that aim at protecting software systems from cyber attacks and threats posed by intelligent adversaries (e.g., hackers or cybercriminals). Whereas Privacy focuses on giving individuals control over their personal information and determining how it is collected, used, stored, and shared. Regulations and standards, such as GDPR, are constantly emerging across different domains to address security and privacy concerns.

When developing chatbots that are used for software engineering purposes, the information and data that are involved are often very sensitive. This may include confidential company data, production code, as well as requirements and specification documents.

During interactions between humans and chatbots, two types of data are involved [2].

First, the prompts in the chats include text messages, voice or even messages. From a security perspective, they must be securely transferred to the chatbot’s hosting server (typically a cloud service) to prevent data leakage. This can be achieved by employing encryption protocols (e.g., SSL/TLS) to protect the confidentiality and integrity of the messages exchanged between the user and the chatbot.

When it comes to privacy, these conversations can be used to retrain the machine learning model powering the chatbot or even shared with third-party services. Consequently, a clear privacy policy is essential, describing how the data is used for retraining purposes and ensuring users have the option to give or revoke their consent.

Second, the user's personal information. Many chatbots, such as ChatGPT and GitHub Copilot, employ user authentication and verification of the users before they can use the chatbot. So, the security of the user's credentials and information when it comes to its processing and storage is crucial.

Furthermore, chatbots can be used to support security. One idea that could be applied to software systems is integrating Chatbots into security monitoring systems to receive and process incident reports from users. For example, if someone notices suspicious activity, they can report it to a chatbot that will analyze the information and alert the appropriate security personnel accordingly.

0.4.2 Quality assurance

Quality assurance encompasses a range of activities carried out throughout the software development lifecycle to ensure that software products meet defined quality standards, reduce defects, and improve customer satisfaction, including assessing the source code for attributes like maintainability, scalability, and readability, as well as verifying requirement documents for clarity, completeness, and alignment with customer objectives.

In relation to my research, chatbots can support the quality assurance in an organization in many ways. One way is using test generative chatbots such as [3] that automatically generate a wide range of test cases, covering different scenarios, inputs, and edge cases. Although the test may not achieve the best coverage, it can uncover some edge cases and make the testing process more efficient. By identifying potential issues or bugs in unexplored areas, these chatbots can enhance the effectiveness of quality assurance efforts. Additionally, chatbots, like other software systems, need to be quality assured. However, there are unique aspects to chatbots that need to be considered for quality assurance. These include testing the accuracy and effectiveness of natural language understanding, testing the conversational flow, and in many cases testing the continuous learning ability of the chatbots. Quality assurance of chatbots comes with significant challenges. An example is handling the ambiguity of the user's input when testing the effectiveness of a response. This opens up for research opportunities in the area of quality assurance for chatbots.

0.5 Future trends of Software Engineering

In the future, software engineering related to chatbots is expected to witness several trends. First of all, there will be a considerable amount of research toward making the chatbot as human-like as possible. This comes in two aspects. Firstly, natural language processing enhances chatbots' language understanding capabilities, enabling them to classify users' intents more accurately and handle complex language nuances. Secondly, there will be a focus on the langugae aspect in the chatbots to make chatbots more human-like by incorporating context awareness, sentiment analysis, and more human-like traits. Additionally, it is expected that a trend in software engineering research will be focusing on ways to interact with chatbots. For instance, multimodal interactions which enable users to engage with chatbots through various interfaces like voice, text, gestures, and AR/VR, enhancing the user experience is expected to be a trend. This requires AI systems that are capable of understanding the users' commands and translating them to the chatbot without requiring them to do further interpretation. Moreover, since chatbots are becoming more common and popular among a wide variety of users, there is a need to conduct research on how to make sure that the chatbots are secure and also to make sure that the user's data privacy is preserved. There are multiple ways in which AI can be used for that, such as anomaly detection on the prompt level. Finally, chatbots are probably going to get integrated into many of the IoT devices that we used on daily bases. Hence, a future trend of integration with IoT devices to allow users to control and interact with their connected devices through chat interfaces, providing a seamless smart home management experience is likely to be a trend.

References

[1] S. I. Ross, F. Martinez, S. Houde, M. Muller, and J. D. Weisz, “The programmer’s assistant: Conversational interaction with a large language model for software development,” in *Proceedings of the 28th International Conference on Intelligent User Interfaces*, pp. 491–514, 2023.

[2] M. Hasal, J. Nowaková, K. Ahmed Saghair, H. Abdulla, V. Snášel, and L. Ogiela, “Chatbots: Security, privacy, data protection, and social aspects,” *Concurrency and Computation: Practice and Experience*, vol. 33, no. 19, p. e6426, 2021.

[3] C. Lemieux, J. P. Inala, S. K. Lahiri, and S. Sen, “Codamosa: Escaping coverage plateaus in test generation with pre-trained large language models,” in *International conference on software engineering (ICSE)*, 2023.