Artificial Intelligence in Software Development Life Cycle

Salwa Shamma

Samah Shamma

Sana Shamma

Somiyah Saad

University of Prince Mugrin

AI 381: Artificial Intelligence I

Section L01

Ms. Mona Algarni

Dec. 10, 2022

Abstract

This report investigates the effects of using artificial intelligence in the cycle of building and developing software. It states the reasons and results for using artificial intelligence and the most important its applications in software development.

Keywords: SDLC, Software, Models, Artificial intelligence, NLP

Artificial Intelligence in Software Development Life Cycle

Introduction

Software engineering and artificial intelligence have grown independently throughout time. While software engineering methodologies are focused with assisting engineers to produce better software in less time, AI approaches enable perception, reasoning, and action. Software engineering tasks have been aided or automated by artificial intelligence approaches, which seek to build software systems that display some aspect of human intellect. Software audits have been successfully used to find flaws in a variety of software artifacts, including requirements, designs, test plans, and source code (Sorte et al., 2015).

This research was carried out in order to investigate the influences of using Artificial Intelligence on software development life cycle.

Analysis Phase

The first stage of software development life cycle is the analysis phase, in this stage the stakeholders of the system are consulted to determine the system's objective, features, boundaries and constraints. After that, they are clearly specified and act as a system specification (Sommerville, 2018). A major part of initializing a software project is requirement elicitation and specification therefore the effectiveness of the coming phases highly depends on the quality of the specified requirements (Sharma et al, 2019).

Reasons to Use AI in Analysis Phase

Engineers and analysts of requirements encounter a variety of difficulties. One of these difficulties is approving good interactions between them and the stakeholders. In the conventional elicitation methods, such as interviews or brainstorming arise the problem of uncertainty and ambiguity between the stakeholder and the analysts which results in

incorrect requirement and such flaw jumps to the later stages of the development process (Sharma et al, 2019).

Example of the Tool Used in Analysis Phase

ScopeMaster is a tool that is concerned with managing the activities of software analysis stage to eliminate the need of rework, assist in delivering more rapidly. ScopeMaster utilizes the power of natural language NL to accomplish analysis procedures in an efficient manner and verify the gathered requirement to detect the ambiguity and inconsistency and more. ScopeMaster recognizes and verifies user stories, assesses software size according to ISO standards, monitors the change of scope, reinforces critical requirements reasoning by generating graphics and a strong detection of anomaly , it can also generate verifiable testcases (Scopemaster, 2022).

The Results of Using AI in the Analysis Phase

As discussed earlier, analysis and requirement specification acts as the basis of the software development lifecycle so by integrating AI techniques in this stage it reduces the needed time in verifying the requirements and user stories and assist in Requirement-traceability by using swarm methodology to trace errors utilizing certain transmissions and to locate the matching documents, they concentrated on text and links in a certain order (Khan & Javed, n.d.).

As claimed by the creatures of ScopeMaster the use of this automated tool by the aid of artificial intelligent techniques the need of reperforming the work is reduced by 20 percent, and it compress the project duration by 5 percent minimum (Scopemaster, 2022).

Design Phase

The second stage is to design the software. In this stage, the designer will review user requirements and work on the initial design of the software, similar to what architects do

when they draw plans for a house before it is built. According to Sharma (2017), the inputs from the users from the previous stage and analysis information are the inputs for this stage. The designer brings down the result from analysing the process and using some diagrams, such as use-case diagrams and class diagrams, to design the software. So, in the design phase, the program requirements written as text are converted and represented in a design model (diagrams), which describes the structure of the software and its component parts. In the advanced stages of modelling, known as "details design," what has been modelled can be directly reversed into code.

Reasons to Use AI in Design Phase

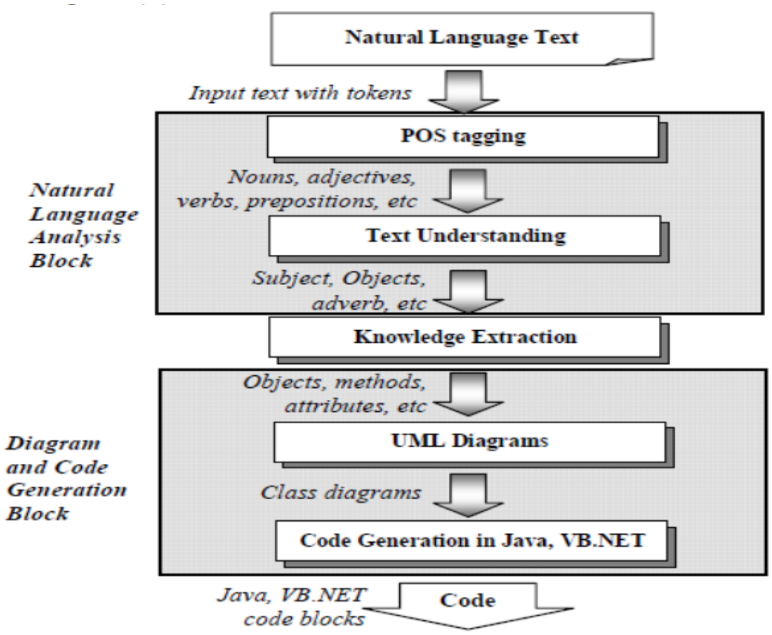
Human mistakes, sometimes occur when the model is generated, some of the requirements may be lost or not considered, and this in turn may lead to incomplete project delivery or customer dissatisfaction because it has lost one of the characteristics that it desires to have in its program (Dawood & Sahraoui, 2017).

Example of The Tool Used in Design Phase

“Subhash and other built a tool that analyses requirement texts and builds model of the processed text represented in the semantic network. Their tool consists of two modules, NL analysis and diagram & code generation. In the first module POS tagging is used to analyze and classify tokens, then the text understanding categorizes text into more further classes, as object, messages, verb, etc. to facilitate class generation process, Knowledge extraction module receives the output of previous phases to extract classes, attributes, and actors. Finally, the UML diagrams will be generated. Based on generated UML diagrams and extracted knowledge Second module generates code in language like java” (Dawood & Sahraoui, 2017, as cited in Subhash et al. 2016, p.45).

Figure 1

Architecture of the designed system



The Results of Using AI in The Design Phase

The main purpose of introducing AI in the design phase is to enhance software quality and minimize design time, cost, and error by reducing human intervention in the design phase through improving the process of generating UML diagrams from requirements using NLP. In addition, artificial intelligence helps speed up the generation of the model by making it do routine work, which gives additional time to designers for more creativity and model creation (Barenkamp et al., 2020; Dawood & Sahraoui, 2017).

But making a system based on artificial intelligence to generate a full design by analyzing the requirements is not possible. The fact that the process of creating designs is a creative process, and this characteristic is difficult to incorporate into computer systems (Meziane & Vadera, 2010).

Implementation Phase

The third stage is to implement the software. In this stage, developers bring the artifacts as inputs, which were done in the previous phases, such as the Software Requirements Specification (SRS) document and design models (diagrams), to life – create an executable program as an output.

There is no standardized procedure that is used in programming; it is an individual activity. Some programmers begin with components they are familiar with and develop them before moving on to less well-known elements. Other coders use the opposite approach, saving the familiar parts for last. After a program is written in particular programming languages. Programmers test the code that they have written. This often uncovers program errors (bugs) that must be removed from the program (Sommerville, 2018). The next section will go deeper into this phase as the next stage of the SDLC, which is testing.

Reasons to Use AI in Implementation Phase

Adopting the automation principle in this phase can be done through different tools. There are different reasons to engage AI in the implementation stage, such as reducing errors, speeding up the implementation process, and handling the critical issue of mismatching between code and diagrams. This mismatching can be generated when a part of the code is modified without reflecting in the diagrams, and vice versa. As a result of this mismatch, documentation will be useless when we revisit it to extend and change. So, we need to automate this process.

Example of the Tool Used in Implementation Phase

One of the automation tools is GitHub Copilot. It is one of the main applications of AI autocomplete suggestions. GitHub Copilot is an AI pair programming that generates

code either by starting to write the code you want to use or by writing natural language comments that describe the code you want to write.

Copilot is a form of an NLP model (GPT-3) that is trained on open-source code. They trained GPT-3, an autoregressive language model, using 175 billion words. Test the effectiveness of the model in a few-shot setting with 10x more parameters than any previous non-sparse language model. Tasks and few-shot demos are provided for all tasks using only text interaction with the model; with no gradient updates or fine-tuning, GPT-3 performs well on numerous NLP datasets. However, there is still a lot of work to be done to improve the model's performance on some of the datasets that it uses (Brown et al., 2020).

The Results of Using AI in the Implementation Phase

There is still argument about whether there is a real need for some automation tools, such as model-driven engineering, that transfer the models to executable programs, in other words, code generators similar to compilers, at this stage as a result of the broad adoption of agile techniques. However, these tools certainly have their strengths and weaknesses. Here’s a roundup of key takeaways. Starting with strength points: Faster coding by eliminating the need to search for a solution using a search engine, and adapting pair programming using an AI. In term of weakness, it takes some time in the maintenance stage since we need to know what these tools do and why they work. And they require a qualified mentor to evaluate these suggestions and decide whether to accept, skip, or ignore them (Scarlett, 2022).

Testing Phase

The fourth stage is software testing. In this stage, the tester will run and execute the system or program to try to find defects or bugs. Moreover, the testing phase is more than

just finding bugs; it also ensures the quality and reliability of software (The application of AI in SE, 2021). The conventional wisdom in the testing field is that testing can reveal the presence of errors, not their absence.

The test phase checks two things: the first situation, the software matches its requirements, which means there is at least one test for every requirement or feature. The second situation, software behaviour correctly by providing groups of inputs or input sequences where the system may produce incorrect computation and data corruption (Summerville, 2018).

Reasons to Use AI in Testing Phase

In the software testing stage, around 40%–50% of resources are consumed, and almost 50% of software development time is spent during it. As a result, software testing is an expensive task, and artificial intelligence is encouraged to get involved in it (application of AI in SE, 2021).

AI in software testing as it is mentioned early helps reducing development time that the software spends in this phase, but that is not all reasons to use it, it just the tip of the iceberg , AI in testing assists also in keeping the code consist specifically when new features are add and new version is developed where testers come with new additions that may break the logic of the previous existing features, so it hard to test everything from beginning with every time we modify or adding, but with AI testing tools, they run only the necessary test case and remove the redundant one (Arnon, 2018).

Example of the Tool Used in Testing Phase

NLP has a significant role in the evolution of software testing. NLP aims to train computers to read or write like humans. One of the NLP applications in an AI-based test tool is Testsigma. Testsigma is a AI cloud-based tool that helps testers test web applications, mobile applications and API's using test cases written in simple English, so

tester doesn't need to learn programming languages to write test cases, so Testsigma depends on NLP in their functionality (Lavanya, 2019).

Testing tools depend on machine learning as well; they can learn about applications under test by collecting data like screenshots, HTML pages, and pages loading time. These data train the algorithm, and when there are errors like slow run time the tools market them as potential issues, so the testers need to check them (Parmar, 2021).

The Results of Using AI in the Testing Phase

The main results of using AI in software testing are enhanced accuracy and reduced test time, cost, and mistakes by reducing human intervention in the test stage through improving the process of generating test cases from natural language to code. By involving artificial intelligence in software testing, the developers spend less time in this stage so they can focus more on other tasks or more on the complexity of AI-based tasks (Dilmegani, 2022).

But the question here is: how about ethical behaviour? How do we test for unethical or ethical behaviour? In a situation where grey areas changes exist How to handle it?

Conclusion

In conclusion, this report has highlighted how AI is used in the software development life cycle (SDLC). Furthermore, it has provided a logical answer to a research question: What are the effects of using AI in the Software Development Life Cycle (SDLC)? AI has a positive impact on each phase, such as reducing the error percentage that can be generated by the change, accelerating software development, and improving accuracy. These tools, on the other hand, require a qualified mentor to evaluate these suggestions and decide whether to accept, skip, or ignore them.

This report discusses AI in each phase, which are the analysis phase, design phase, implementation phase, and test phase, in terms of the reasons that motivate the use of AI in the phases, an example of a tool that can be used in this stage, and the impact of using AI in that process.

AI in the software development life cycle is still a research topic and it is a promising field. We encourage anyone who is interested in connecting these different fields to conduct their own research in these areas and come up with proposals to implement some tools that can be used in these SDLC phases.

Reference

Axelrod, A. (2018). *Complete Guide to Test Automation: Techniques, Practices, and Patterns for Building and Maintaining Effective Software Projects* (1st ed.). Apress.

Barenkamp, M., Rebstadt, J., & Thomas, O. (2020). Applications of AI in Classical Software Engineering. AI Perspectives, 2(1). https://doi.org/10.1186/s42467-020-00005-4

Batarseh, F. A., Mohod, R., Kumar, A., & Bui, J. (2020). The application of artificial intelligence in software engineering. *Data Democracy*, 179–232. https://doi.org/10.1016/b978-0-12-818366-3.00010-1

Brown,T,. Amodei,D,. Ryder, L., Alec 1, McCandlish,S,. Kaplan,J,. Agarwal,S., Askell,A. Sastry,G., and Clark,J. "ArXiv.org e-print archive," 2020. [Online]. Available: https://arxiv.org/pdf/2005.14165v4.pdf. [Accessed: 09-Dec-2022].

Dawood, O. S., & Sahraoui, A.-E.-K. (2017). From requirements engineering to UML using Natural Language Processing – survey study. European Journal of Engineering Research and Science, 2(1), 44. https://doi.org/10.24018/ejers.2017.2.1.236

Development is fast-paced, so should be Testing to keep up with the accelerated delivery cycles. But Testing has not evolved. (2019, May). Smart Test Automation Using NLP. Retrieved December 10, 2022, from https://dev.to/lvnya_c/smart-test-automation-using-nlp-h9b

Dilmegani, C. (2022, September 30). Complete Guide on AI in Software Testing: Benefits & Challenges. *AIMultiple*. https://research.aimultiple.com/ai-testing/

Khan, R. Ali, & Javed, S. (n.d.). *Use of Artificial Intelligence in Software Development Life Cycle*. University of Management And Technology.

Meziane, F., & Vadera, S. (2010). Artificial Intelligence in Software Engineering. Advances in Computational Intelligence and Robotics, 278–299. https://doi.org/10.4018/978-1-60566-758-4.ch014

Parmar, A. (2021, November 25). *How AI and ML can help Test Automation?* https://www.linkedin.com/pulse/how-ai-ml-can-help-test-automation-ashish-parmar/

Scarlett, R. (2022, July 26). Why Use GitHub Copilot And Copilot Labs: Practical Use Cases for the AI Pair Programmer. DEV Community. https://dev.to/github/why-use-github-copilot-and-copilot-labs-practical-use-cases-for-the-ai-pair-programmer-4hf4

Scopemaster. (2022, November 27). *Solutions*. ScopeMaster. https://www.scopemaster.com/wp-content/uploads/2021/07/ScopeMaster-Brochure-2021-v2.pdf

Sharma, M. (2017). A study of SDLC to develop well engineered software. International Journal of Advanced Research in Computer Science, 8(3), 520- 523. https://eds.b.ebscohost.com/eds/pdfviewer/pdfviewer?vid=4&sid=d83d5411- fef7-4433-85e9-58101eb1d343%40pdc-v-sessmgr02

Sorte, B., Joshi, P., & Jagtap, V. (2015). Use of Artificial Intelligence in Software Development Life Cycle: A state of the Art Review. *International Journal of Advanced Engineering and Global Technology*, *03*(03). https://www.researchgate.net/publication/274254538_Use_of_Artificial_Intelligence_in_Software_Development_Life_Cycle_A_state_of_the_Art_Review

Sommerville, L., Software engineering. Boston: Pearson Education Limited, 2018.