Applications of Artificial Intelligence Techniques to Software Engineering

WORD COUNT:
1456

ABSTRACT

Software development is the process of conceiving, specifying, designing, programming, documenting, testing, and bug fixing involved in creating and maintaining applications, frameworks, or other software components[1]. Software development at its current state is mainly a human interacted activity. Software development requires a high level of knowledge of multiple domains (problem & programming) and the ability to merge and combine different aspects and techniques into a single solution.

This document means to audit the techniques and strategies which have been created and used by Artificial Intelligence in the respect to software engineering. Specifically, this document centers around techniques created or which are being created in AI that can be used in tackling issues related with software engineering.

INTRODUCTION

The software frameworks that we build up are getting significantly more complex in terms of functional and non-functional prerequisites. Standards must always be maintained as low quality can have great negative effect on the system applications. Furthermore, the expense of creating such frameworks usually outweighs all other costs. In the last few decades, we have made major breakthroughs in applying AI methods to software development which has increased the production of projects and publications.

As you are reviewing this document you will most likely come across three main questions. What exactly is software engineering? What exactly is Artificial Intelligence? And what makes software engineering so receptive to AI techniques and strategies?

Software engineering is the process of making software applications using software development techniques. Software engineering usually consists of two parts analysis and synthesis. The breakdown of something into smaller and simpler versions to better understand it is know as analysis. Constructing large structures from smaller broken down versions of said structure is know as synthesis.[2] Analysis and synthesis are the polar opposites of each other. Knowing this we can confirm that most problem solving techniques consist of both analysis and synthesis. Figure 1. demonstrates this logic.

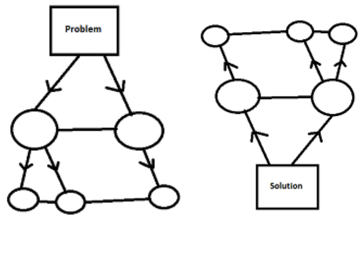

Figure 1. (left) Analysis of Problem (right) Synthesis of Solution

Artificial Intelligence is an area of computer science that endeavors to cause machines to do what humans previously did. This concept of creating intelligent machinery has been dreamed of for centuries, but that dream is slowly becoming a reality in the current times.[3]

ARTIFICIAL INTELLIGENCE & SOFTWARE ENGINEERING

Software engineering depends on two types of understandings: programming (syntax, control, data structures and understanding of when to use what) and domain. When a certain software is being developed the language in which it is being developed must be linked to the domain. In the conversion both languages must be formal. Therefore, a conversion is necessary, the methods of conversion require AI intervention.

Modelling, generation of alternate design and analysis are all a part of software engineering that brings about problem breakdown. Methods are required to choose a suitable design in order to analyse and measure the breakdowns. This type of activity is manageable by AI.[4]

A sub field in AI can be utilized when translating informal descriptions into formal descriptions of prerequisites. The AI's task isn't to directly translate but instead to aid the user as conversion is not a fully automated straight forward process. Although, the AI can create queries and illustrations of what the user is scripting in natural language. The feedback from these AI generated queries generate what the system will form. An example of this is present in database management systems (ex. MySQL) where the user can use natural English to form code and queries.

This AI sub field incorporates; Language Conception (interaction of machine and human using symbols i.e. text), Sound Synthesis(interaction of machine and human using sound) & Voice Transmission(interaction of machine and human using vocal input).

The process of software development usually spans from requirements specification to testing the program. Various kinds of understandings are required at each stage for example the programming understanding at the coding writing stage and the domain understanding at the domain stage. At both of the stages, plan and code scripting, exist a cycle: mistake acknowledgement and revision. Experience demonstrates that mistakes can happen at any phase. Blunders when writing code may happen as a result of flawed planning. Such mistakes are generally costly to address.

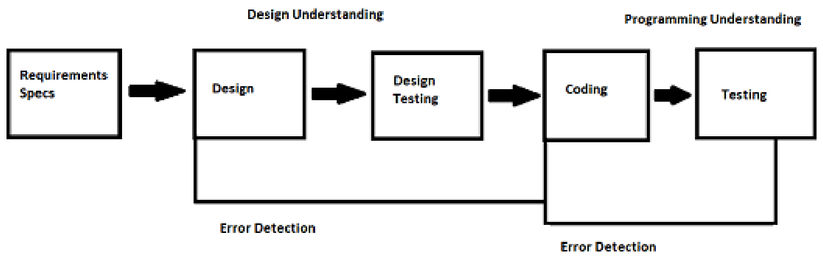

Figure 2. Software Development Phases

One technique is to automatically translate the first stage to the last testing stage. Taking the problem in continuation and making changes at the specification stage accomplishes this. The machine translates the user input (requirements) into code. This process has many benefits; it significantly reduces the cost, mistakes can be detected and amended easier and quickly at the requirement phase, all changes and adjustments only need to be made at the requirement stage. One downside to this is that it is hard to implement onto larger programs for guaranteed success.

To produce a framework in software engineering, one may discover anther framework with comparative requirements. The plan of the initial framework would be changed until it turns into a sensible plan for the given problem. Despite the fact that this cycle looks achievable, it has not been shown in software engineering to any substantial degree. Constraint Propagation is a technique that is offered by AI, which provides rise to a truth maintenance system that can be utilized for plan making. Decision making and carrying out plans take place at different levels of the development phase. Thus, employing analogical analysis(a product of logical analysis) and problem solving is used for comparison with which another problem can be solved with that solution.

A basic problem of software engineering is the long delay between the requirements specification and product delivery [5]. This long process causes changes in requirements before product release. Therefore, its critical to have an automated framework using AI, which can accept the requirements and complete all the multiple levels of translation to codes.

Additionally, phase independence becomes an issue, it means that any decision at one phase must be fixed for the next. This causes the developer to change and recode whenever faced with a change in design. Although, when using AI this issue is avoided as the technique used is automated resulting in reusable code. Thus, when using AI the part the isn’t changed remains as it is. This method cannot function properly without using a constraint propagation technique.

TECHNIQUES

Due to the constant evolving nature of software when you complete your code the prerequisites would have changed (due to the long phases of development). Therefore, you would need design by experimentation whose practicability lies within AI. Tools successfully implemented into automated programming include;

- Language Feature: intends on making data structures flexible, also known as late binding. These data structures are not completed into a particular data implementation structures. Hence, quick same versions of code are created resulting in overall more efficient code as it can be changed easily. A further useful feature is the packaging of data, thus giving emergence to object-oriented programming

- Program Browsers: these browsers examine various segments of a code that are still being created or modified, potentially to make changes, subsequently forestalling the need for a normal text editor. The program comprehends the structures and revelations of the program and can zero in on the part of the program that is of interest.

- Meta Programming: is a natural language developed concept, meta programming generates executable lisp code using automated parser generators and interpreters. Its benefit is found in user interfaces, data transformations and modelling of transition sequences.

CONCLUSION

Albeit an exceptionally formal hypothesis has been examined, AI in software engineering still has limitations and constraints, it can also in some cases be illogical. With ideas well outlined, the issue is in the synthesis of larger problems. Therefore, appropriate cases will be required to be distinguished where these developments are efficient. Likewise, when automating program development using simple concepts must be language independent. One of the upgraded language components is object-oriented programming which lessens the prerequisites to code phase. Object oriented concept has now been completely refined and developed which is also now fixated in a number of object-oriented languages such as Python, C++, and among others.

References:

[1] wikipeida.org/wiki/software_development

[2] https://ncu.libguides.com/writingresources/synthesis#:~:text=Synthesis%20and%20Analysis%3A%20combine%20and,a%20theory%2C%20discussion%2C%20or%20interpretation

[3] Computers and Computing Houghton Mifflin College Publishers 1989

[4] Shari Lawrence Pfleeger, Software Engineering: theory and Practice (Prentice Hall Publishers, Upper Saddle River, New Jersey, USA) 1998.

[5] Data Processing and Information Technology (10th Edition) Letts Educational Publishers, London 1996.