**AI in (Design?) Phase of SE - title**

**\*References no properly included!!!**

**Abstract**

The areas of Software Engineering and Artificial Intelligence are among the most sought after fields in the computer science community.. This paper assesses the current scenario of implementing artificial intelligence in the field designing in software engineering, and attempts to offer a viable explanation to support the conclusion of the assessment.

**General Terms:** TBM

**Keywords:** TBM

**Introduction** (put in context of design phase)

In this day and age, traditional software design has its limits. In traditional SE, the approach towards building a software is manual in nature, which makes it prone to human errors such as oversight and is also inefficient when we consider complex projects. (Assumption Statement/Problem Statement)

Therefore, in order to minimize human error and increase efficiency of software design we can implement AI techniques, mainly in the stages of design, testing, GUI testing, strategic decision making, automatic code generation and intelligent programming assistants. (Assumed Solution)

Hence, we must look at the prospects of automation/AI in various phases of software design and to deal with related problems, which are the reliability and accuracy of automation and how much control does it take away from a software developer. (Problems with the solution).

This survey paper compares related works regarding this and how they either tackle this problem or suggest tackling this problem.

(Discussion related to division of section of paper)

**Literature Review/Existing Work**

(Structure of this section may be wrong)

1. **Artificial intelligence and software engineering: Status and future trends**

This paper discusses the various commonalities between the fields of AI and SE, current status and upcoming trends. Intersections between AI and SE are discussed, especially in the fields of Agent-Oriented Software Engineering, Knowledge Based Engineering Systems, Computational Intelligence and KDD and Ambient Intelligence. It also gives a short description about AI and software engineering, and how the possibilities of interaction with one another through various common points of contact.

1. **Artificial intelligence and systems engineering. Prospects for Artificial Intelligence**

This paper discusses the differences between the software engineering process and AI, and the hostilities that exists in software engineers towards using automation in developing a software. It further discusses the contribution of AI to the field of SE in fields like automatic programming and some projects like REFINE and Programmer’s Apprentice (an MIT project) and the problem areas of SE where AI can be implemented like requirements analysis and definition, process modelling and process support and project planning.

The author states one of the reasons for the existing hostility towards AI is that AI researchers are themselves software engineers, and therefore their method to integrating AI in software engineering follows a typical and traditional approach which precedes the advancements that AI and its endless possibilities can bring to the field of software engineering.

1. **Software engineering using artificial intelligence techniques: Current state and open problems**

This paper uses the terminology defined IEEE 12207 standards for software engineering to describe the development processes of requirements analysis, software architecture design and coding and testing and AI techniques that can be implemented along with them. The paper also talks about open problems in SE that can be solved using AI so as to improve the process of SE in this day and age where software are getting more modular and complex to develop.

1. **Ways of applying artificial intelligence in software engineering**

This paper deals with a taxonomy called AI in SE Application Levels or AI-SEAL that classifies 15 papers from previous editions of the RAISE workshop. It considers the context in which AI is being applied, I.e. “when” and “on what” AI is being applied. After classification the papers also assigns levels of automation to the papers.

1. **Artificial intelligence and techniques in software engineering**

This paper gives a brief overview of SE and expert systems in artificial intelligence and how expert systems can be used to automate the programming process and code generation via the use of genetic programming.

This paper also shows the absence of risk management in AI based systems due to the way they work.

1. **From user requirements to UML class diagram \*important paper**

Translation of user requirements into design diagrams is a daunting task for a designer as that person has to translate textual requirements into a diagrammatic (UML) form. This paper gives a basic overview of existing systems regarding and how they convert the user requirements using NLP into UML diagrams..

They develop their own tool DC Builder and presents in the paper a diagrammatic representation and a heuristic rule set in implementing NLP on user requirements.

Finally they evaluate and compare the various tools mentioned and developed with one another.

1. **Generating UML Diagrams from Natural Language Specifications**

The extraction of UML diagrams from textual requirements by requirements engineers is a daunting task and the time and effort spent on this justifies a tool to automate this process which brings the author to propose a tool called RAPID which uses NLP in an efficient manner to extract design diagrams from input textual user requirements.

The methodology states various NLP technologies, algorithms and rules to extract class information from the textual requirements.

1. **Textual Requirements for UML Diagram Extraction by using NLP**

This paper talks about the ambiguity of textual requirements and the usage of NLP and domain ontology to generate UML diagrams from the said textual requirements.

This paper also talks about existing systems and then approaches the problem using their own method called RAUE to extract UML diagrams from textual requirements.

1. **Class Diagram Extraction from Textual Requirements Using Natural Language Processing (NLP) Techniques**

This paper looks at existing systems which translate textual requirements into UML diagrams. After that the author proposes his own approach for RACE which is an improvement on the TCM system.

The methodology of the RACE system is discussed witch it’s algorithms and rules to identify classes, attributes and relationships.

The author concludes the paper stating RACE being an advanced approach towards extraction of UML diagrams from textual requirements using a human-centered UI and what the system did not support.

1. **ATGen: Automatic Test Data Generation using Constraint Logic Programming and Symbolic Execution**

The paper introduces an automatic test data generator called ATGen. ATGen is based on constraint logic programming and symbolic execution. Developing softwares include a number of testing methods. The main technique that is currently used in the industry is dynamic software testing where the software is executed using test data. Dynamic testing can be performed using automatic tools, which are - automation of administrative tasks, automation of mechanical tasks and automation of test generation tasks.

ATGen is a prototype testing tool implemented using the ECLiPSe constraint logic programming environment and consists over 5000 lines of commented Prolog code. The current area of application of ATGen is the automatic generation of test data to achieve 100% decision coverage for programs written in SPARK Ada. In decision testing, the aim is to test all decision outcomes in the program. The aim is to generate a test data suite achieving 100% decision coverage. However, initial results of ATGen has proved that at the time this research was conducted it was impossible to experiment with ATGen using industrial SPARK Ada code as such code was not usually made available even for research due to its safety critical aspects. Also, the test results show that ATGen shows wide performance variation between successive runs, particularly for programs with loops. The initial results of ATGen were promising and the overall efficiency of the algorithm is under improvement.

1. **CM Builder: A Natural Language-based Case Tool**

This paper describes an approach towards extracting UML diagrams from natural language using NLP via a tool called CM Builder. CM Builder is a graphical CASE tool that does surface analysis of text to propose candidates for class, attributes and relationship and domain independent semantic analysis to automatically extract the candidates and finally represent them via UML diagrams.

The tool also includes capacity to evaluate it’s candidate classes. The author also states benefits of object oriented analysis of the tool and the scope of improvement for it.

1. **Engineering: Will the twain ever meet?**

This session explored the reasons for the lack of impact in four important areas in which AI has been expected to significantly affect real world Software Engineering. The session approached the failures of AI in software engineering, looking at the matter through a common cause- reliance on isolationist technology and approaches, rather than upon creating additive technology and approaches that can be integrated with other existing capabilities.

The isolationism has been manifested in several areas, inn essence, the market has rejected the isolationist and egocentric approach to implementing AI in software engineering, and that the whole system should be developed and executed in generalized workstations and PCs.

1. **Regular Expressions from Natural Language Specifications: Are We There Yet?**

Recent state-of-the-art approaches automatically generate regular expressions from natural language specifications. Given that these approaches use only synthetic data in both training datasets and validation/test datasets, a natural question arises: are these approaches effective to address various real-world situations? To explore this question, in this paper, a characteristic study on comparing two synthetic datasets used by the recent research and a real-world dataset collected from the Internet, and an experimental study on applying a state-of-the-art approach on the real-world dataset is conducted. The study results suggest the existence of distinct characteristics between the synthetic datasets and the real-world dataset, and the state-of-the-art approach (based on a model trained from a synthetic dataset) achieves extremely low effectiveness when evaluated on real-world data, much lower than the effectiveness when evaluated on the synthetic dataset.

1. **Issues in Developing UML Diagrams from Natural Language Text**

This paper looks at previous work done in the field of converting natural language requirements into design diagrams and points us the various issues we can encounter in this process.

The author points out the issues in the context of software engineering, then in the context of natural language and finally proposes a methodology to solve these problems to get proper quality UML diagrams.

1. **Artificial intelligence and software engineering: a tutorial introduction to their relationship**

The paper introduces artificial intelligence to software engineers and conversely, software engineering to artificial intelligence workers. It further highlights the contrast between the two

fields and further accentuates their differences in the problems that they attempt to solve, their methodologies and the tools and techniques. The author strongly believes that the work of software engineers and artificial intelligence workers are similar, and the fusion of the two fields is essential for Computer Science as a field to move forward, however, the author also acknowledges the ridge that exists between the two communities. The paper does not however provide the ‘*how*’ on which AI and software engineering can come together, other than details of the core components of both fields.

The paper acknowledges that to move forward with ‘user-friendliness’, the system needs to allow natural language input and output, and natural language processing is one of the most researched

areas in AI. In other words, a system needs AI to further improve its usability.

Conversely, AI is in need of proper models to address its problems. These models are already evident in conventional systems, and it is clear that AI systems need some standardization in

order to widen their applications.

Thus, there exists a requirement for both fields to merge on some common ground.

**Results and Analysis of Existing Work**

From the study done above a few things are apparent. Firstly, when we consider design diagrams made by a software engineer, three factors come into play-

* Ambiguity of customer defined requirements
* Complex nature of the project due to increasing complexity of technology and also the size of the project
* Possibility of human error when designing those systems

This results in the problem of defining the accuracy of a system, be it designed by a person or by an AI based tool, as we have no point of references, making it imperative to create a dataset to compare any diagram to measure its accuracy.

Secondly, when we consider AI in system design, due to it’s automatic nature we encounter the problem of a system engineer not having enough freedom over the creation of design diagrams. Hence, we need to carefully define the level of automation of such tools.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Support** | **CM-Builder** | **LIDA** | **GOOAL** | **NLOOML** | **DC-Builder** |
| Classes | Yes | User | Yes | Yes | Yes |
| Attributes | Yes | User | Yes | Yes | Yes |
| Methods | No | User | Yes | Yes | No |
| Associations | Yes | User | Semi-NL | No | Yes |
| Multiplicity | Yes | User | No | No | No |
| Aggregation | No | No | No | No | Yes |
| Generalization | No | No | No | No | Yes |
| Instances | No | No | No | No | No |

Table I: Evaluation of Tools’ functionalities (Adapted from paper [])

The table above shows us the comparison of the availability of features of various tools. And we can infer (like the paper) that there are very few tools which fully extract relationships from textual requirements, which is due to the ambiguity of textual requirements.

**Conclusion**

This paper aims to specify the problems that can be encountered when developing design diagrams by a software engineer in terms of a customer’s system requirements, the difficulty of measuring the accuracy of a generated design diagram, the problem of specifying the level of automation and finally we look at tools an the various features that thy support.

Hence we can com to the conclusion that further work needs to be done in order to fully implement AI in system design.

**References**