Generating Software Development Process using Artificial Intelligence Methods

R.M.Jayasree[1] M.Sreemaa[2]
[1]P.G. Student [2]Associate Professor
[1,2]Department of Computer Science & Engineering
[1,2]Surya Group of Institutions, Vikiravandi, Villupuram, Tamil Nadu India

Abstract— In this paper discusses the mechanism of artificial intelligence applied to software engineering process. Employing artificial intelligence in this manner, salvages the wastage of time in case of marketing, and also increases the quality of software systems. This paper relates AI techniques to software engineering processes specified by the IEEE 12207 standard of software engineering. The paper is driven by the activities and tasks specified in the standard for each software engineering process. This discussion brings the state of the art of AI techniques closer to the software engineering process, and highlights the open research problems for the research community. Surveyance of AI related to software engineering process is also described based on various procedures adapted to the specified standard.

Key words: Automated Software Engineering, Artificial Intelligence Techniques, software architecture

I. INTRODUCTION

Research in applying artificial intelligence techniques to software Engineering have grown tremendously in the last two decades. The software intensive systems we develop these days are becoming much more complex in terms of the number of functional and nonfunctional requirements they need to support. The impact of low quality can also have a catastrophic impact on the mission of these systems in many critical applications. Moreover, the cost of software development dominates the total cost of such systems. The AI techniques attempt to automate or semi-automate these tasks and produce optimal or semi-optimal solutions in much less time.

A. Overview of the Paper:

- IEEE 12207 standard.
- Description of AI techniques employed.

B. IEEE 12207 standard for Information Technology:

This standard establishes a framework for software life cycle processes, and provides well-defined terminology to be used by all the stakeholders. It defines processes, activities, and tasks that are to be applied during the acquisition of a system that contains software, a stand-alone software product, and software service. The standard covers the tasks for the overall the life cycle phases during the supply, development, operation, maintenance and acquisition of software products. The standard also defines four project organizational processes that include the management process. Eight other supporting processes are defined in the middle layer to support the primary processes and the management process in the top layer.

Fig. 1: IEEE 12207 development process

II. DEVELOPMENT PROCESS

This process focuses on requirements analysis, architecture design, coding, and testing. The standard provides a flow of the development process activities and associated documents. This includes the Software architecture and requirements allocation description (SARAD) document, software requirements specification document (SRS), software architecture description (SAD), software interface design description (SIDD). This is followed by the software coding and testing activities.

III. REQUIREMENT ENGINEERING (RE)

Requirements are first expressed in natural language within a set of documents. The main activities of this phase are requirements elicitation, gathering and analysis and their transformation into a less ambiguous representation.

Some of the problems encountered in this phase are:

- Requirements are ambiguous.
- Requirements are incomplete, vague and imprecise.
- Requirements are conflicting.
- Requirements are volatile.
- There are communication problems between the stakeholders.
- Requirements are difficult to manage.

IV. ONTOLOGIES

Ontologies are developed by many organizations to reuse, integrate, and merge data and knowledge and to achieve

interoperability and communication among their software systems.

A. Intelligence Computing for Requirements Engineering:

Some of the systems developed using Computational Intelligence (CI) techniques are established below:

The SPECIFIER system takes input as an informal specification of an operation where the pre and post-conditions are given as English sentences. Computational linguistics is used to analyze textual scenarios, to identify where actors or whole actions are missing from the text, to fill the missing information, and to generate a message sequence chart (MSC) including the information missing from the textual scenario. A Collaborative and situational tool called MUSTER, has been designed and developed for requirements elicitation workshops. This provides more regional approach.

B. Software Architecture Design:

One of the most important problems that the software engineer face, is to develop quality architecture from the requirements model. AI techniques uses quality attributes to define a goodness function over the space of possible architectures. Genetic Algorithms (GAs) are used to search the space of possible hierarchical decompositions of a system.

C. Software Coding & Testing:

1) Coding:

Software engineers can apply AI techniques to help automate or assist the programming process.

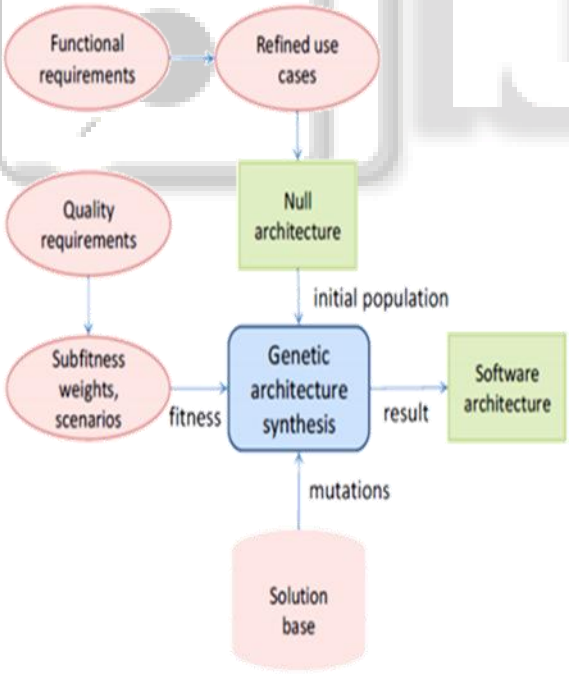

Fig. 2: Evolutionary Architecture generation from requirements.

a) Use of AI to help assist the programming process:

The main idea here is to create an expert system to assist software engineers during software development. This is called the Programmer's Apprentice Project. The Programmer's Apprentice should have the capability of interacting with the human programmers exactly the same way as human assistants would, thereby hopefully increasing the productivity of the human programmers. At first, the Apprentice would only be able to handle "the simplest and most routine parts" of programming.

b) Use of AI to help automate the programming process:

The idea here is to have a completely automated program synthesis. This is done by having human specialists to write a complete and concise specification of the desired software; so that, a system can generate "functions, data structures, or entire programs" directly from the specifications. There are many possible AI technologies that could be applied.

- Constraint programming is another AI technique that is applied in software engineering.
- PTIDEJ(Pattern Trace Identification, Detection and Enhancement in Java). is an automated system designed to identify micro-architectures looking like design patterns in source code.
- Search Based Software Engineering (SBSE) focuses on representing aspects of Software Engineering as problems that may be solved using meta-heuristic search algorithms developed in AI.

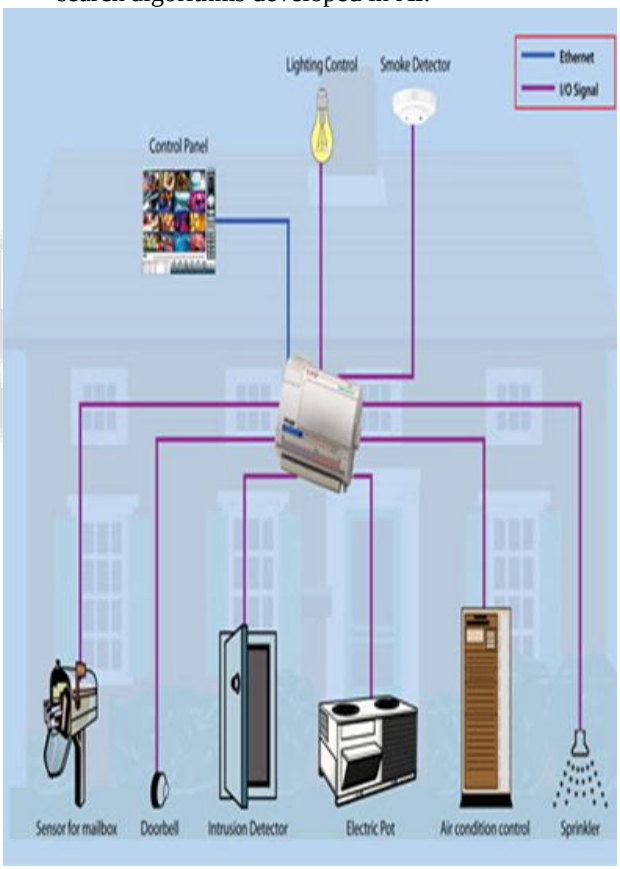

Fig. 3: Real Time automation of simple home installations & its Programs.

V. TESTING

Software testing remains an expensive task in the development process and one of the main challenges concerns its possible automation. AI techniques can play a vital role in this regard. One of these techniques are constraint solving techniques.

- Prolog based expert system is used to generate test data based on the conditions.

- AI planning was used for testing distributed systems and for the generation of test cases for graphical user interfaces.
- Fuzzy logic is another AI technique that is applied in software testing to manage the uncertainty involved in this phase of software development.

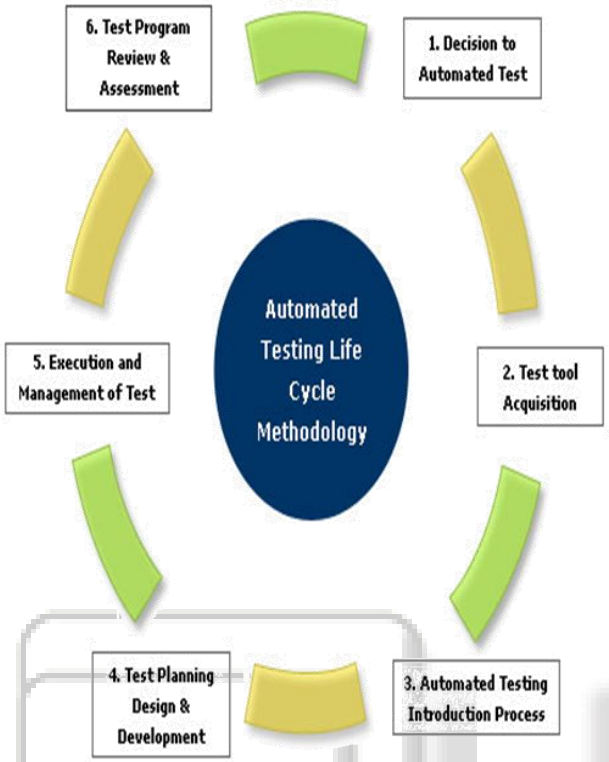

Fig. 4: Testing process implemented through AI

VI. CONCLUSIONS

In this paper, the evaluation of applying AI techniques is acquired, to solve some of the important problems facing the software engineer. Recent trends in the development activities requirements engineering, software architecture design coding and testing processes have concurrently used AI techniques for its implementation and execution.

The upcoming issues due to the manual software development and AI generated software processing are also reviewed. The Advancement in software development achieved through the usage of AI is highly credited and discussed in our paper.

REFERENCES

[1] Abbott, R. J. (1983). Program design by informal English descriptions.
[2] Bering, C. A., & Crawford, M. W. (1988). Using an expert system to test a logistics information system. In Proceedings of the IEEE National Aerospace and Electronics Conference Dayton.
[3] Vadera (Eds.), Artificial Intelligence Applications for Improved Software Engineering Development: New Prospects.
[4] Falbo, R. A., Guizzardi, G., Natali, A. C., Bertollo, G., Ruy, F. F., & Mian, P. G. (2002), Towards semantic software engineering environments.
[5] Sommaruga, L.; Perri, A.; Furfari, F.; DomoML-env: an ontology for Human Home Interaction. Proceedings of SWAP (2005)
[6] Furfari, F.; Sommaruga, L.; Soria, C.; Fresco R. DomoML: The definition of a standard markup for interoperability of human home interactions. Proceedings of the 2nd European Union symposium on Ambient Intelligence (2004)
[7] Bonino, D.; Corno, F.; Dogont-ontology modeling for intelligent domotic environments. 7th International Semantic Web Conference (2008) pp. 790-80
[8] Valiente, P. and Lozano-Tello, A. "Control Model of Domotic Systems based on Ontologies". It will be presented at the 2nd International Conference on Agents and Artificial Intelligence. Valencia, Spain. January 2010.
[9] Hepp, M.; De Leenheer, P.; de Moor, A.; Sure, Y. (Eds.) Ontology Management. Semantic Web, Semantic Web Services, and Business Applications, Vol. 7. Springer 2008

