Applications of Various Artificial Intelligence Techniques in Software Engineering

Dipen Saini

Department of Computer Science , DAV college, Jalandhar(Punjab), INDIA

er.dipensaini@gmail.com

ABSTRACT

Software engineering is the principles which are used in the development of the software. Software development is a very long, complex process and it's a human oriented activity. This paper discusses the concept of Artificial Intelligence and how its various techniques can be used in software engineering i.e. how artificial intelligence techniques are related to software engineering? Various Artificial Intelligence techniques are discussed in this paper, which can be used to improve many of the software development activities, with which quality of the software can be increased.

Keywords: Software Development, Software Engineering, Artificial Intelligence, Artificial Intelligence Techniques.

1. INTRODUCTION

When you have started reading this research paper, these three questions might have stricken your mind.

- What is Software engineering?
- What is Artificial Intelligence?
- How Artificial Intelligence is related to Software engineering?

Software Engineering is the act of adopting engineering principles that can be used for efficient and effective development of high quality and mostly very large software system. The goal is to support software engineer and managers in order to develop better software faster with intelligent tools and method. In this act principles of analysis and synthesis observed. *Analysis* is the process of decomposing something into components or modules with a view to understand the individual component.

Synthesis, the reverse of analysis, is the process of putting together of small building blocks to build a large structure. (Emrich M.L, et al, 1988).

Thus any problem solving techniques must have two parts:-

- Analyzing the problem to determine its nature.
- Synthesizing a solution based on the analysis.

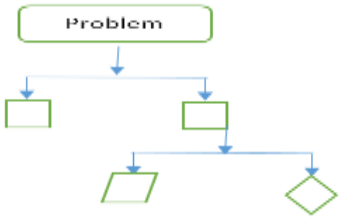

Fig. 1(a) analysis of the problem to determine

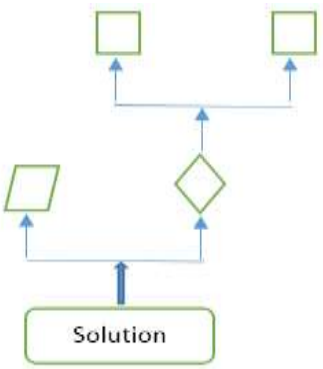

Fig. 1(b) a synthesis of a solution based on the analysis.

Artificial intelligence is that field of computer usage which attempt to construct computational mechanism for activities that are considered to require intelligence when performed by humans.

Artificial intelligence focus on creating machine that can engage in behave that humans considered intelligent. Artificial Intelligence is concerned with the study and Creation of computer systems that exhibit some form of intelligence and attempts to apply such knowledge to the design of computer based systems that can understand a natural language or understanding of natural intelligence. One element in AI methodology is that progress is sought by building system that performs **Synthesis** before **Analysis**. (Wachsmuth, 2000**)**

2. ARTIFICIAL INTELLIGENCE IN SOFTWARE ENGINEERING

Software engineering is the introduction of formal engineering principle to the creation and production of the software. Software engineering basically deals with the development of the software. Development of software is a long process which goes through various phases and requires an executable code. Human beings have been writing a code for the development of the software for a long time and no machine can yet do better.AI is the process of creating intelligent machines that perform tasks done by human beings. Artificial intelligence technique can be used to improve many of the software development activities. Hence, there is significance potential for improving all phases of SDLC (Software development life cycle).

3. RELATED WORK

According to (Katz and Zimmerman-Gal, 1981), an advisory system is developed to assist human programmers in selecting data structures. This system is currently in use in an educational setting.

The XPLAIN system incorporated techniques for explaining the behavior of a program by examining the expert knowledge from which the program was derived. The XPLAIN experiments were done in the domain of digitalis therapy (W. Swartout, 1983).

The DESIGNER project used psychological modeling and protocol analysis techniques to characterize the behavior of human algorithm designers. This characterization was the basis of attempts to automate the design of certain algorithms [D. Steier and E. Kant, 1985].

RML is a formalism for expressing software requirements using knowledge representation techniques. It has been used to describe the requirements for several small information systems. (Borgida *et al*, 1985).

According to (Jonathan, 2002), automated programming is a solution to phase independence, which results in reusable code. Some of the techniques and tools that have been used in an automated programming environment are:

- Language Feature: Language feature is a technique which uses the concept of late binding. It means making the data structure flexible. Another important language feature is the object oriented programming which is a package of data and procedures in an object.
- Meta Programming: This concept is developed in natural language processing and, it uses automated parser generators and interpreters to generate executable lisp codes.
- Program Browsers: Program browsers looks at different portions of the code to make changes, thus obviating the need for ordinary text ordinary.
- Automated Data structuring: It is a concept in which we map the high level specification of data structures into the implementation structure.
-

4. ASPECTS OF SOFTWARE ENGINEERING AND ARTIFICIAL INTELLIGENCE

Software engineering categorizes two kind of knowledge programming knowledge (for e.g. - data structure construction, control structure, programming language syntax and how to combine and choose them) and domain knowledge (e.g. - concepts, theories and equations characterizing the particular domain).a domain has unique properties which must be mapped to the programming language in which soft. Is being developed, thus it require a conversion from one knowledge to another (i.e. from domain knowledge to programming knowledge or from domain to programming language).In this we require some method from these types of conversion, which require some techniques in AI.

The traditional view of software development process begins at the requirements specification and ends at testing the software. At each of the stages, different kinds of knowledge (design knowledge at design stage and programming and domain knowledge at the coding stage) are required. At each of the two stages: design and coding exist a cycle: error recognition and correction. Experience shows that errors can occur at any stage of development. Error due to coding may occur because of faulty design.

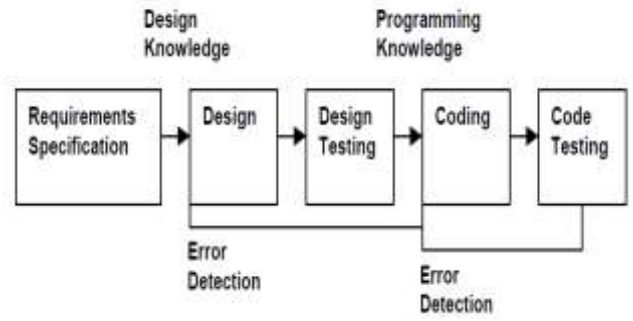

Fig. 2 The traditional software development process.

A basic problem of software engineering is the long delay between the requirements specification and the delivery of a product. This long development cycle causes requirements to change before product arrival. There is the problem of phase independence of requirements, design and codes in software development process. Phase independence means if we have made any decision any one level of the software development process, become fixed for the next level. Thus the coding team is forced tore code whenever a change occurs in the requirement.

The term *Artificial intelligence* (AI) refers to the intelligence of machines and the fields of scientific and engineering research that strive to develop that intelligence. The design of AI systems focuses on the creation of intelligent entities that are capable of perceiving their environment and reacting to various stimuli.

*According to Wachsmuth*it is not the aim of AI to build intelligent machines having understood natural intelligence but to understand natural intelligence by building intelligent machines" A further sub-division adapted from *Richter and abecker* of AI into sub-fields, methods and techniques is shown in figure 3.

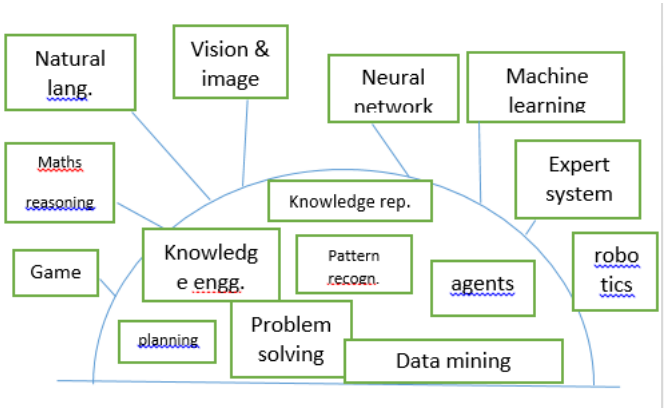

Fig. 3 AI fields, methods and techniques.

Intersections between AI and SE:

Today, methods and techniques from both disciplines support the practice and research in the respectively other research area. Figure 4 represent the current research areas in artificial intelligence, software engineering as well as common areas between them. The common research area between artificial intelligence are- ambient intelligence, computational intelligence agents and knowledge based software engineering.

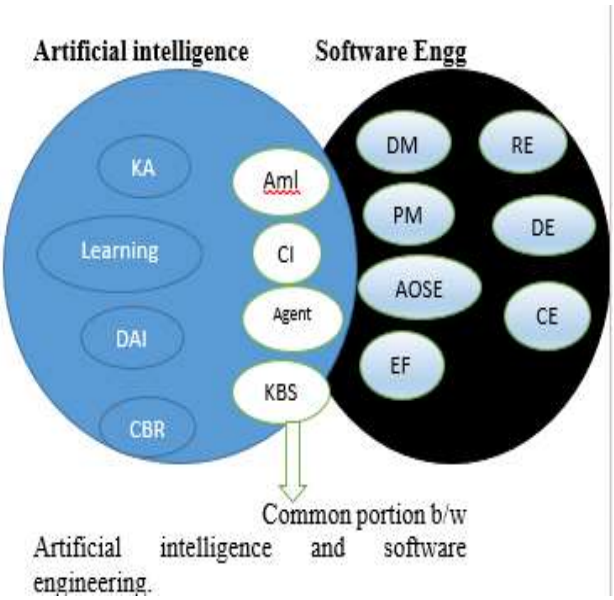

Fig. 4 Research Areas in AI and SE and common between the two.

Ambient intelligence help us making a sensitive, adaptive and reactive systems that are informed about the user's needs, habits and emotions in order to support them in their daily work (*Da Costa and Punie Y, 2003*).There are several AI research areas for the development of smart algorithms for Aml applications (*Verhaegh et al, 2004*) e.g., user profiling,

context awareness, scene understanding (*Aarts E. H. L., 2003*), or planning and negotiation tasks. Research from the SE side is concerned with model-driven development for mobile computing (*Da Costa and Punie Y, 2003*), the verification of mobile code (*Roman et al., 2002*), the specification of adaptive systems, or the design of embedded systems *(Basten et al,2003*).

Computational intelligence has many techniques like neural networks, evolutionary algorithms, or fuzzy systems are increasing applied and adapted and adapted for specific SE problems. They are used to estimate the progress of projects to support software project management (Brandt M and Nick M, 2001), for the discovery of defect modules to ensure software quality, or to plan software testing and verification activities to minimize the effort for quality assurance (khoshgoftaar T.M., 2003).

Software Agents are typically small intelligent systems that cooperate to reach a common goal. These agents are a relatively new area where research from KI and SE intersects. From the AI side the focus in this field lies on even more intelligent and autonomous systems to solve more complex problems using communication languages between agents in SE agents are seen as systems that need more or less specialized formal methods for their development, verification, validation, and maintenance.

5. ARTIFICIAL INTELLIGENCE TECHNIQUES

Expert system development: -Expert system use knowledge rather than data to control the solution process. Knowledge engineers build systems by eliciting knowledge from experts, coding, that knowledge in an appropriate form, validating the knowledge, and ultimately constructing a system using a variety of building tools.

The main phases the expert system development processes are:-

- Planning
- Knowledge acquisition and analysis
- Knowledge design
- Code
- Knowledge verification
- System evaluation

Planning phase involves feasibility assessment, resource allocation, and task phasing and scheduling Requirements analysis. Knowledge acquisition is the most important stage in the development of ES. During this stage the knowledge engineer works with the domain expert to acquire, organize and analyze the domain knowledge for the ES. The goal of knowledge analysis is to analyze and structure the knowledge. Gained during the knowledge acquisition phase. After knowledge analysis is done, we enter the knowledge design phase. At the end of design phase, we have Knowledge definition, detailed design, and decision of how to represent knowledge decision of a development tool. Consider whether

It supports your planned strategy, internal fact structure, Mock interface. Coding this phase occupies the least time in the Expert System Development Life Cycle. It involves coding, preparing test cases, commenting code, developing User's manual and installation guide.

Knowledge-based techniques in AI can be used to modify traditional software development approach. one strategy is to automatically translate from the requirement specification to program testing. The user only provides the requirement and machine build using does the translation into the codes. (Engr.FarahNaazRaza, 2009)

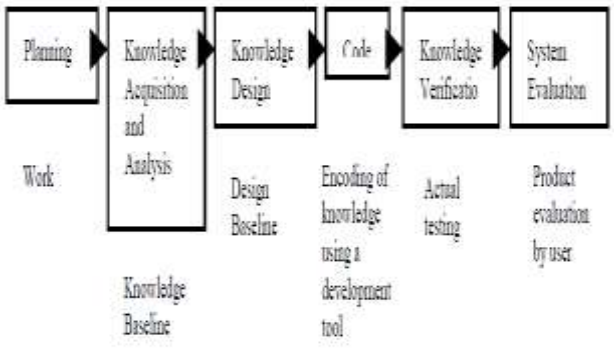

Fig. 5 Expert System development

Richter defines three different levels as essential for describing knowledge base system. The cognitive layer (human oriented, rational, and informal), the representation layer (formal, logical) and implementation layer (machine oriented, data structure and programs).

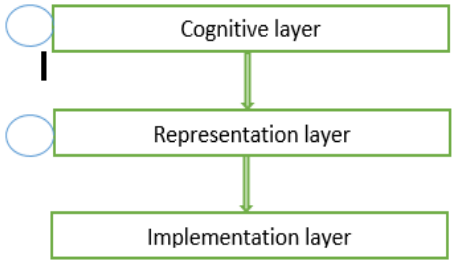

Fig. 6 the three levels of knowledge based systems

The AI technique that handles phase independence problem is automated programming which results in reusable code. Thus, when a change is made in the design, that part of the design that does not change remains unaffected. Thus, automated tools for system redesign and reconfiguration resulting from a change in the requirements will serve a useful purpose.

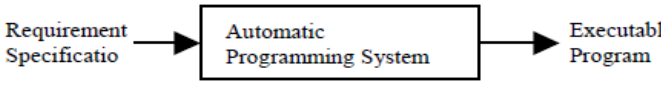

Fig. 7 Automatic Programming System (APS)

Neural networks

It has been widely and successfully used for problems that require classification given some predictive input features. They therefore seem ideal for situations in software engineering where one needs to predict outcomes, such as the risks associated with modules in software maintenance (khoshgoftaar andlanning, 1995), software risks analysis (Neumann, 2002) and for predicting faults using object oriented metrices (Thwin and Quah, 2002).

Genetic Algorithms

In the areas of software development, Shan et al, 2002 utilize Genetic programming to evolve functions for estimating software effort. Two target grammars were adopted for the functions that allowed use of a range of mathematical functions (e.g., exp, log, sqrt) as well as a conditional expressions.

Fuzzy logic

Fuzzy logic is a form of many-valued logic or probabilistic logic; it deals with reasoning that is approximate rather than fixed and exact. is another AI technique that is applied in software testing to manage the uncertainty involved in this phase of software development (Nand et al, 2007).

6. The Role of AI Techniques in Software Development Activities

The Development Process

In this particular AI techniques will focus on the tasks related to requirements analysis, architecture design, coding, and testing. (Hany H Ammar et al,2010)

1.) *Software requirements analysis-*

i. Requirement Engineering (RE):

Requirements are first expressed in natural language within a set of documents. These documents usually represent "the unresolved views of a group of individuals and will, in most cases be fragmentary, inconsistent, contradictory, not prioritized and often be overstated, beyond actual needs". The main activities of this phase are requirements elicitation, gathering and analysis and their transformation into a less ambiguous representation.(Young,2003)

Problems arising during this phase can be summarized as follows:

- 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

In the following section some of the techniques used in the requirement engineering field are discussed:

ii. Processing Natural Language Requirements NLR

A framework to translate specifications written in NL (English) into formal specifications (TELL) was introduced

but the system was not implemented, it set the foundations for future systems.NL2ACTL system was introduced, which aims to translate NL sentences, written to express properties of a reactive system, to statements of an action based temporal logic. Another system, FORSEN system was developed and aim was to translate NL requirements into the Formal specifications language VDM. This system allowed the detection of ambiguities in the NL requirements. (Saeki et al,1989)

Some of the examples of the systems that have attempted to produce OO oriented models from NL Requirements are discussed below: (Fentechi, 1994)

A general framework for the automatic development of OO models from NL requirements using linguistics instruments was introduced. A Large-scale Object-based Linguistic Interactor Translator Analyzer (LOLITA) NLP system was used to develop the NL-OOPS which aims to produce OO specifications from NL requirements, an approach was developed that linked the linguistic world and the conceptual world through a set of linguistic patterns. Another system was developed named as the Class-Model Builder (CM-Builder), a NL based CASE tools that builds class diagrams specified in UML from NL requirements documents.(Mezaine,1994)

iii. *RISK MANAGEMENT*

The Risk Management process is a method of identifying risks in advance and establishing methods of avoiding those risks and /or reducing the impact of those risks should they occur. The process of risk management begins during the analysis phase of software development life cycle. However, the actual process of managing risks continues throughout the product development phase.

The given Figure displays the steps of the risk management process. Formally, articulated, risk management process consists of three steps:

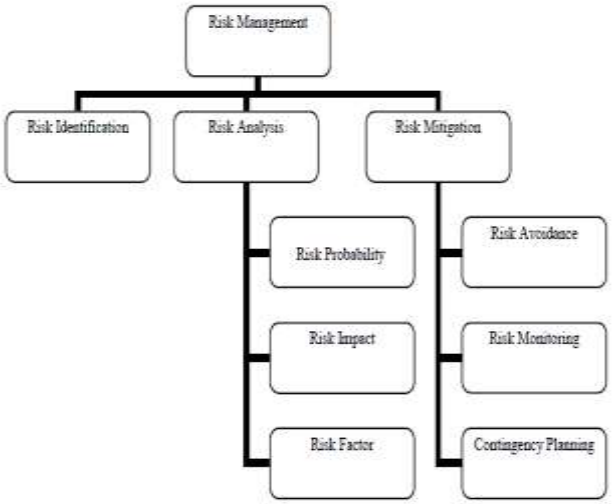

Figure 8 Risk management process

AI based systems are free from risk management strategies because of automated programming techniques making data structures flexible. Automatic programming is the generation of programs by computer, usually based on specifications that are higher-level and easier for humans to specify than ordinary programming languages.

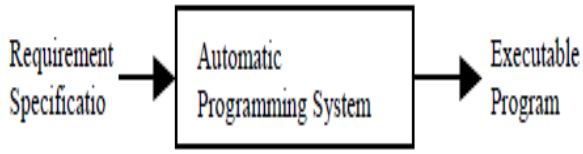

Figure 9 Automatic Programming System (APS)

The goal is to make the specification smaller, easier to write, easier to understand (closer to application concepts), Less error-prone better than programming languages

2.) *Software architecture design*

One of the most important problems facing the software engineer is to develop quality architecture from the requirements model. In this section a description on recent work on software architecture design using AI techniques are described. Developing the software architecture starts by defining a hierarchy of subsystems and components with

allocated responsibilities from the information provided by the requirements and analysis models. AI techniques uses quality attributes to define a goodness function over the space of possible architectures. Some of the most common quality attributes of architecture design used in developing the architecture are modularity, complexity, modifiability, understandability (or clarity), and reusability. Modularity is usually connected to the concept of coupling and cohesion, where designers strive for a modular design by developing the architecture using loosely coupled and highly cohesive subsystems and components. In an earlier work on using AI techniques for software architecture development, Robyn Lutz used Genetic Algorithms (GAs) to search the space of possible hierarchical decompositions of a system. A fitness function using information theoretic metric capturing the data coupling and control coupling between components. The quality attribute used for the fitness function is related to the complexity and modularity of the produced architecture. Later on, focus was on Product Line Architectures (PLAs) where variation points are explicitly defined to enhance reusability and modifiability of reference architecture that can be used to instantiate a family of architectures.

3.) *Software coding and testing*

Techniques learned from AI research make advanced programming much simpler, especially with regard to information flow and control as a result of advances in knowledge representation. In the following we focus on the AI techniques used in supporting the tasks of coding and testing.

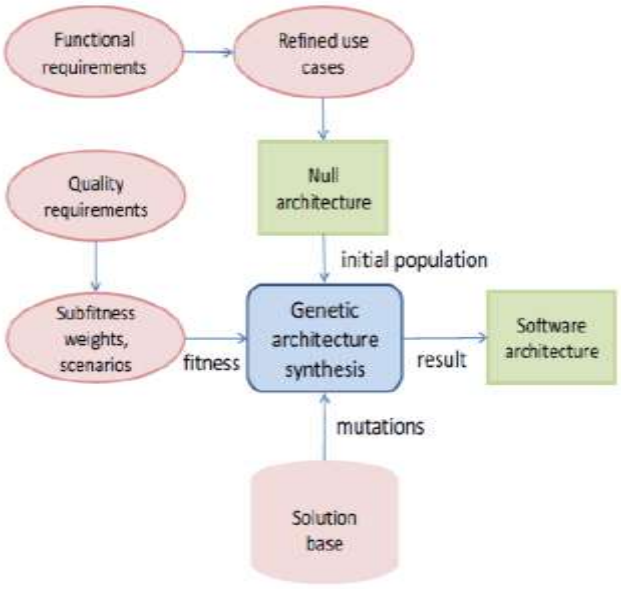

Figure 10 Evolutionary Architecture generation

a) Coding:

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

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. There was one proposal, 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. As time progresses and research continues, the Apprentice should be able to deal with more complicated tasks. The human programmers will still be necessary to implement code of a 'tricky' nature (such as abstract reasoning or to better cater human preferences). (Partridge, 1991)

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 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. Analogical reasoning in software reuse can be used. The idea is to find a system with similar requirements and modify it. Although this process looks feasible, it has not been demonstrated in software engineering to any great extent.(Hewet,1994)

Constraint programming is another AI technique that is applied in software engineering. Constraint programming has been, for example, used to design the PTIDEJ system (Pattern Trace Identification, Detection and Enhancement in Java. PTIDEJ is an automated system designed to identify micro-architectures looking like design patterns in object oriented source code. A micro-architecture defines a subset of classes in an objected oriented program. The main interest of PTIDEJ is that it is able to provide explanations for its answers. This is really interesting since coding and software engineering is often considered a form of art and where fully automated systems are not always appreciated by potential users (or programmers).

Search Based Software Engineering (SBSE) is an emerging research topic that focuses on representing aspects of Software Engineering as problems that may be solved using meta-heuristic search algorithms developed in AI. SBSE is the reformulation of software engineering tasks as optimization problems. One of the optimization and search techniques that can be used are genetic algorithms. Genetic algorithms are used for automatic code generation by optimizing a population of trial solutions to a problem. The individuals in the population are computer programs.

b) 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. In the context of mutation testing, much attention has been devoted to the use of constraint solving techniques in the automation of software testing (Constraint-based testing). ATGen, for example, is a software test data generator based on symbolic execution and constraint logic programming for ADA programs. There are many other ways how AI techniques can support the testing process]. One of the earliest studies to suggest adoption of knowledge based system for testing, which describe a Prolog based expert system that takes a COBOL program as input, parses the input to identify relevant conditions and then aims to generate test data based on the conditions.

Fuzzy logic is another AI technique that is applied in software testing to manage the uncertainty involved in this phase of software development. (Nand, 2007)

7. CONCLUSION

Software engineering helps us to build a software product but by following the software engineering principles, the development of product takes very much time. The quality of the product can be increased by using AI techniques in the software development. By using AI based systems with the help of automated tool or automated programming tool we can eliminate risk assessment phase saving our time in software development, and building an effective product. Because of Artificial techniques in Software Engineering, we can reduce the development time in software development. Coding phase in software development process can be changed into Genetic Code, which is the main concern in the development of software.

8. Future Work

One of the most difficult problems in the development of software is the problem of transforming requirements into architectures. So, research should be done on this thing and efficient product line architectures and service-oriented architectures using AI techniques should be developed.

Test case generation is notoriously hard in software testing. Recent work should make progress towards the ultimate goal of fully automated test case design generation.

More work should be done on coding and testing part in software development using AI techniques to have a high quality product in less time and well mannered way.

References

[1] A. Borgida, S. Greenspan, and J. Mylopoulos.," Knowledge representation as the basis for requirements specifications".1EEE Computer, 18:82-91, April 1985.

[2] Da Costa O and Punie Y., Science and Technology Roadmapping: Ambient Intelligence in Everyday Life (Ami@Life):Unpublished IPTS working paper, 2003.

[3] E. Kant,"Understanding and automating algorithm design", IEEE Transactions on Software Engineering, 11(11):1361-1374, November 1985.

[4] **Fantechi**, A., Gnesi, S., Ristori, G., Carenini, M., Vanocchi, M., & Moreschini, P. (1994). Assisting requirement formalization by means of natural language translation. Formal Methods in System Design, 4(3), 243–263.

[5] Hany H Ammar, Walid Abdelmoe, and Mohamed Salah Hamdi,"Software Engineering Using Artificial Intelligence Techniques: Current State and Open Problems", 2010.

[6] **Hewett**, Micheal, and Rattikorn Hewett (1994). 1994 IEEE 10th Conference on Artificial Intelligence for Applications.

[7] Jonathan Onowakpo Goddey Ebbah ,"Deploying Artificial techniques in Software Engineering",Department of Computer Science University of Ibadan, Nigeria, 18 March-2002

[8] **Meziane**, F. (1994). From English to Formal Specifications. PhD Thesis, University of Salford, UK.

[9] M.L. Emrich, A. Robert Sadlowe, and F. Lloyd Arrowood (Editors)," Expert Systems And Advanced Data Processing": Proceedings of the conference on Expert Systems Technology the ADP Environment (Elsevier-North Holland, New York, New York, USA) 1988.

[10] **Meziane**, F. (1994)." From English to Formal Specifications", PhD Thesis, University of Salford, UK.

[11] **Partridge**, Derek, ed. (1991). Artificial Intelligence and Software Engineering. New Jersey: University of Exeter, 1991.

[12] Roman G.C., Picco G.P., and Murphy A. L., "Software Engineering for Mobility: A Roadmap," presented at Future of Software Engineering Track of 22nd ICSE, Limerick, Ireland, 2000.

[13] **Saeki**, M., Horai, H., & Enomoto, H. (1989),"Software development process from natural language specification". In Proceedings of the 11th international Conference on Software Engineering. (pp. 64-73), Pittsburgh, PA.

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

[15] S. Katz and R. Zimmerman.," An advisory system for developing data representations"in Seventh International Joint Conference on artificial Intelligence, pages 1030-1036, Vancouver, British Columbia, Canada, August 1981.

[16] Verhaegh W.F.J., Arts E.H.L., and Korst J.,Algorithms in ambient intelligence, vol.2. Boston: Dordrecht, ISDN: 140201757X, 2004.

[17] W. Swartout. Explain,"A system for creating and explaining expert consulting systems" Artificial Intelligence Journal, 21(3):285-325, September 1983.

[18] **Young,** R. R. (2003). The requirements Engineering Handbook. Norwood, MA: Artech House Inc.

