RESEARCH ARTICLE OPEN ACCESS

A Survey on Using Artificial Intelligence Techniques in the Software Development Process

K. Hema Shankari[1] , Dr. R.Thirumalaiselvi[2]

[1] Research Scholoar, Bharath University and Assistant professor, Department of Computer Science, Women's Christian College, Chennai.

[2] Research Supervisor and Assistant professor, Department of Computer Science, Govt. Arts College (Men) (Autonomous), Nandanam, Chennai.

ABSTRACT

Software engineering and artificial intelligence are the two important fields of the computer science. Artificial Intelligence is about making machines intelligent, while Software engineering is knowledge –intensive activity, requiring extensive knowledge of the application domain and of the target software itself. This study intends to review the techniques developed in artificial intelligence from the standpoint of their application in software engineering. The goal of this research paper is to give some guidelines to use the artificial intelligence techniques that can be applied in solving problems associated with software engineering processes.

The aim of this paper is to find out the exact AI technique is likely to be fruitful for particular software development process

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

I. INTRODUCTION

Software development process is a very complex process that, at present, is primarily a human activity. Programming in software development, requires the use of different types of knowledge: about the problem domain and the programming domain. It also requires many different steps in combining these types of knowledge into one final solution. There are various techniques in artificial intelligence (AI) from the standpoint of their application in software engineering that can be deployed in solving problems associated with software development processes. 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. Many Software products costs can be attributed to the ineffectiveness of current techniques for managing this knowledge, and Artificial Intelligence techniques can help alleviate this situation.

II. LITERATURE SURVEY: AI TECHNQIUES

Hany M Ammar, Walid Abdelmoez and Mohamed Salah (2012) in their paper on the Current state and open problems in the Software Engineering using artificial intelligence discuss about how the artificial intelligent technique such as KBS,CBR, Fuzzy logic and automated programming tool help to overcome the problems associated in the traditional software development. There are certain open problems such as SBST requires further research

Mark Harman (2011) discuss about the three boards areas of AI techniques such as SBSE, Fuzzy , probabilistic method , classification learning and prediction help the software engineering community and also about the challenges ahead in AI for SE.

Farid Meziane, Sunil Vadera (2010) discuss about the current developments and future prospects for Artificial Intelligence in software Engineering. Testing and the other phase of the software development.

Farah Naaz Raza (2009) in the paper "Artificial Intelligent technique in Software engineering (AITSE) " explains about by using AI based systems with the help of automated tool or automated programming tool we can eliminate risk assessment phase by saving the time in software development and also AITSE reduce the development time in the software development.

Parveen Ranjan Srivastava and Tai-hoon Kim (2009) discuss about the application of Genetic Algorithm in Software Testing and this paper clearly says that the GA is used for the improvement of the software testing efficiently.

Mark Harman and Bryan F. Jones (2001) presented paper on the Search-based software engineering and in this paper explains about the search-based techniques could be useful for the development of software measures.

Prince Jain (2011) discusses about the interaction between Software Engineering and Artificial Intelligence and in this paper he explains about the reasons for which AI techniques are needed for the software engineering process.

Jonathan Onowakpo Goddey Ebbah (2002) presented paper on Deploying Artificial Intelligence Techniques in Software Engineering in this paper intends to review the techniques developed in artificial intelligence (AI) from the standpoint of their application in software engineering. In particular, it focuses on techniques developed (or that are being developed) in artificial intelligence that can be deployed in solving problems associated with software engineering processes.

Jyoti tewari,Swati arya, Prem narayan singh (2013) presented paper on Approach of Intelligent Software Agents in Future Development discuss about the intelligent behavior of the intelligent agent predicting the future development in a software.

Dr. Nachamai.M, Senthil Vadivu and Tapaskar (2011) in the paper enacted software development process based on agent methodologies discuss about the how agent based software development have weight factors more than traditional software models.

Seth Hock, (1989) in the paper Computers And Computing explains artificial intelligence on the other hand is a domain of computer science that attempts to make machines perform tasks that hitherto done by human beings .

SURVEY ON AI TECHNIQUES IN 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.

a) Coding:

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

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. Booch (1986) explains the NL description to data types while describing the Object-Oriented analysis and design method.

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.

Closely related to analogical reasoning techniques is Case-based reasoning (CBR). CBR is based upon the premise that similar problems are best solved with similar solutions. CBR is argued to offer a number of advantages over many other knowledge management techniques. For program synthesis retrieval from component repositories and the reuse of successful past Experience is important. As an example, one application of CBR technology was to support the reuse of software packages within Ada and C program libraries.

The idea of experience reuse, the most ambitious form of CBR-supported reuse, is closely aligned with what is called Experience Factory. This field is also known as Organizational Learning, researches methods and techniques for the management, elicitation, and adaptation of reusable artifacts from software engineering projects. An Experience Factory is based upon a number of premises such as a feedback process, appropriate storage of experience, and support of reuse and retrieval.

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. Since the seminal work of Offut and De Millo (1991) 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 a knowledge based system for testing was by Bering and Crawford (1988) who 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.

A more active area of research since the mid-1990s has been the use of AI planning for testing. An AI planner could generate test cases, consisting of a sequence of commands by representing commands as operators, providing initial states, and setting the goal as testing for correct system behavior. AI planning was also used for testing distributed systems and for the generation of test cases for graphical user interfaces.

Stuart Russell and Peter Norvig, (1994) in their paper Artificial Intelligence: A Modern Approach explain the Analysis is the process of breaking something into pieces or components with a view to understanding the individual components .

A study by Kobbacy, (2007) has shown that the use of genetic algorithms for optimization has grown substantially since the 1980s. This trend is also present in their use in testing, with numerous studies aiming to take advantage of their properties in an attempt to generate optimal test cases. Bertolino (2007) presents a useful

framework for summarizing the challenges that are faced in addressing the problems of ensuring that systems are fit for purpose, suggesting further research on: (i) developing a universal theory of testing, (ii) fully automatic testing, (iii) design to facilitate testing and (iv) development

of integrated strategies that minimize the cost of repeated testing . Wappler and Wegener (2006) acknowledge that using a fitness function as the primary means of avoiding illegal sequences is not efficient. Instead they propose a novel use of Genetic Programming (GP), which aims to learn

functions or programs by evolution. The authors in for example, used genetic algorithms for testing object oriented programs where the main aim was to construct test cases consisting of a sequence of method calls.

Nand, S., Kaur, A., Jain S. (2007).in the paper Use Of Fuzzy Logic In Software Development. Issues in Information Systems. Explains about the use of fuzzy logic in software testing to manage the uncertainty involved in this phase of software development.

SURVEY ON AI TECHNIYQUES IN PLANNING AND PROJECT EFFORT ESTIMATION

Good project planning involves many aspects: staff need to be assigned to tasks in a way that takes account of their experience and ability, the dependencies between tasks need to be determined, times of tasks need to be estimated in a way that meets the project completion date and the project plan will inevitably need revision as it progresses. AI has been proposed for most phases of planning software development projects, including assessing feasibility, estimation of cost and resource requirements, risk assessment and scheduling. This section provides pointers to some of the proposed uses of knowledge based systems, genetic algorithms, neural networks and case based reasoning, in project planning and summarizes their effectiveness.

Knowledge Based Systems

There have been several studies that adopt this assumption and aim to capture this experience in a Knowledge Based System (KBS) and attempt to utilize it for planning future software development projects. One of the earliest studies to suggest adoption of a Knowledge Based System (KBS) for testing was by Bering and Crawford (1988) who describes 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. Sathi, Fox & Greenberg (1985) argue that a well defined representation scheme, with clear semantics for the concepts associated with project planning, such as activity, causation, and time, is essential if attempts to utilize KBS for project planning are to succeed. Hence, they develop a representation scheme and theory based on a frame based language, known as SRL (Wright, Fox, & Adam, 1984).Their theory includes a language for representing project goals, milestones, activities, states, and time, and has all the nice properties one expects, such as completeness, clarity and preciseness. Surprisingly, this neat frame based language and the semantic primitives they develop have been overlooked by others and appear not to have been adopted since their development. Gupta, Bastani,Khan & Yen (2004) take advantage of the goal oriented properties of Means-Ends planning by defining potential system actions as operators so that generating tests becomes equivalent to the goal of finding a plan from the current state to specified unsafe or near unsafe states.

Memon, Pollack & Soffa (1999) argue that human generation of test cases for graphical user interfaces requires enumeration of a large number of possible sequences of user actions, making the process inefficient and likely to be incomplete. Instead, as with the above studies, they propose the use AI planning methods, since once the possible actions are specified using operators, a planner can generate tests since it is capable of finding a sequence of actions to achieve a goal from an initial state. Similarly, T. Menzies (2001) that aim to utilize a KBS approach for project management, such as the use of production rules and associative networks which seemed promising at the time have not been widely adopted. Boardman, J. T., & Marshall, G. (1990) explains about the knowledge- based architecture for project planning and control. When considering whether to adopt a KBS approach, the cost of representing the knowledge seems high and unless this can be done at a level of abstraction that allows reuse, one can imagine that it is unattractive to software developers who are keen and under pressure to commence their projects without delay.

Neural Networks

Neural networks (NNs) have 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 and software risk analysis (Neumann, 2002) and for predicting faults using object oriented metrics (Thwin & Quah, 2002). Karlsson and Ryan (1997) , Khoshgoftaar & Lanning (1995) explains the neural network approach. The study by Hu, Chen, Rong, Mei & Xie (2006) is typical of this line of research. They first identified the key features in risk assessment based on past classifications such as those presented by Wallace and Keil (2004) and further interviews with project managers. They identified a total of 39 risk factors which they grouped into 5 risk categories: project complexity, cooperation, team work, project management, and software engineering. These were reduced to 19 linearly independent factors using principal component analysis (PCA). Projects were considered to have succeeded, partially failed, or failed. In their experiments, they tried both the use of a back propagation algorithm for training and use of GAs to learn networks, using 35 examples for training and 15 examples for testing. The accuracy they obtained using back propagation was 80% and that with a GA trained NN was over 86%, confirming that use of NNs for predicting risk is a worthy approach, though larger scale studies are needed.

Genetic Algorithms

There have been numerous uses of genetic algorithms for project scheduling in various domains (Cheng & Gen, 1994; Hindi, Hongbo, & Fleszar, 2002; Hooshyar, Tahmani, & Shenasa, 2008; Yujia & Chang, 2006; Zhen-Yu, Wei-Yang, & Qian-Lei, 2008, Briand, Labiche, & Shousha, 2005;). A survey of their application in manufacturing and operations management can be found in (Kobbacy, Vadera, & Rasmy, 2007; Meziane, Vadera, Kobbacy, & Proudlove, 2000). These typically formulate project planning as a constraint satisfaction problem with an objective that needs optimisation and, which is then transformed into a form suitable for optimisation with a GA.

A study by Kobbacy, Vadera and Rasmy (2007) has shown that the use of Genetic Algorithms (GAs) for optimization has grown substantially since the 1980s and this growth has continued while the use of other AI technologies has declined. This trend is also present in their use in testing, with numerous studies aiming to take advantage of their properties in an attempt to generate optimal test cases (Baresel, Binkley, Harman, & Korel, 2004; Baudry, Fleurey, Jezequel, & Le Traon, 2002a, 2002b; Briand, Feng, & Labiche, 2002; Briand,

Labiche, & Shousha, 2005; T.Menzies(2001)& Emrich & Lylod (1988) Wappler & Wegener, 2006) .In the area of software development, Shan, McKay, Lokan & Essam (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. The approach was tested on data consisting of 423 software development projects characterized by 32 attributes (e.g. such as intended market, requirements, level of user involvement, application type, etc) from the International Software Benchmarking Standards Group (www.isbsg.org.au) with roughly 50% used for training and 50% used for testing. The results of this study show that the approach performs better than linear and log regression models. An interesting finding of the study was that although the most accurate functions discovered by GP utilized similar parameters to the traditional estimates, a key difference was that it adopted non-linear terms involving team size. Creating a good assignment of staff to tasks and producing schedules is critical to the success of any software development project. Yujia & Chang (2006) show how it is possible to utilize GAs to produce optimal schedules and task assignments. Their proposal involves a two part chromosome representation. One part includes the assignment of individuals to tasks and another involves representing the topological ordering of the tasks in a way that ensures that the offspring

generated using the cross-over operator remain valid schedules. The fitness function is obtained by utilizing a systems dynamics simulation to estimate expected task duration given a particular chromosome. The results of their experiments suggest that this is a promising approach, though further work on how to utilize GAs in practice when schedules change is still needed. An important part of developing an optimal schedule that meets a target completion date is the trade-offs that may occur. For example, attempts at increasing quality can result in increasing cost and possibly compromising completion time but
perhaps increasing user satisfaction. Increasing resources on tasks increases the local cost but may result in early completion, higher quality and reduction of overall cost. Hooshyar, Tahmani & Shenasa (2008) propose the use of GAs to optimize schedules to take account of such trade-offs. They represent a schedule by a chromosome consisting of the activity duration and which is ordered based on their dependency. In their experiments, they utilize the standard mutation and two-point cross-over operators and adopt a fitness function that includes the cost and duration. The experimentation is carried out on projects consisting of 10, 20 and 30 activities and concludes that although the well known algorithm due to works well for small scale problems, GAs may be more effective for larger scale problems. Ryan (2000) explains about the Automatic re-engineering of software using genetic programming and Siemens(1971) explains the CPM time-cost tradeoff algorithm which are very useful.

Case Based Reasoning

It can be argued that successful project planning and management is heavily based on experience with past cases. It is therefore surprising that there are few studies that propose the use Case Based Reasoning (CBR) for project planning of software development. One of the few exceptions is the study by Yang and Wang (2009), who explore the combined use of CBR and data mining methods for project planning. They use a structured - representation for cases, called Hierarchical Criteria Architecture (HCA), where projects are described in terms of the customer requirements, project resources and keywords describing the domain. The use of HCA enables different weights to be adopted when matching cases, allowing greater flexibility depending on the preferences of the project manager. Given a new project, first similar new cases are retrieved. Then, data mining methods, such as association rule mining, are used to provide further guidance in the form of popular patterns that could aid in project planning. In a trial, based on 43 projects, Yang & Wang (2009), show how the combined use of CBR and data mining can generate useful information, such as "the duration of project implementation was about 26 days and 85% of projects of projects were completed on time", which can be used to provide guidance when planning a similar project.

Ian Somerville in the book Software Engineering (2000), Roger S. Pressman, in Software Engineering: A Beginner's Guide (1998) explains about Software engineering is the act of adopting engineering principles in software development. In this act, the principles of analysis and synthesis are observed

III. TECHNIQUES AND TOOLS OF AUTOMATED PROGRAMMING

Because of the evolutionary nature of software products, by the time coding is completed, requirements would have changed (because of the long processes and stages of development required in software engineering): a situation that results in delay between requirement specification and product delivery. There is therefore a need for design by experimentation, the feasibility of which lies in automated programming. Some of the techniques and tools that have been successfully demonstrated in automated programming environments include:

• Language Feature: this technique adopts the concept of late binding (i.e. making data structures very flexible). In late binding, data structures are not finalized into particular implementation structures. Thus, quick prototypes are created which result in efficient codes that can be easily changed. Another important language feature is the packaging of data and procedures together in an object, thus giving rise to object-oriented programming: a notion that has been found useful in environments where codes, data structures and concepts are constantly changing. Lisp provides these facilities.

- Meta Programming: this concept is developed in natural language processing (a sub field of AI). It uses automated parser generators and interpreters to generate executable lisp codes. Its use lies in the modeling of transition sequences, user interfaces and data transformations.
- Program Browsers: these look at different portions of a code that are still being developed or analyzed, possibly to make changes, thus obviating the need for an ordinary text editor. The browser understands the structures and declarations of the program and can focus on the portion of the program that is of interest.
- Automated Data Structuring: this means going from a high-level specification of data structures to a particular implementation structure.

When systematic changes need to be made throughout a code, it is more efficient and controllable to do it through another program (i.e.,

program update manager) than through a manual txt editor. For instance, a change in program X may be required whenever h is being updated by b-1 under the condition that b is less than C. Assume that a program W makes a systematic change in all such places. If another program makes a change in W, then any program changed by W also must be updated. Thus, program update managers propagate changes. Because of this ability, program update managers are useful when prototypes need to be developed quickly.

IV. NEED FOR ARTFICIAL INTELLIGENCE TECHNIQUES IN SOFTWARE ENGINEERING

Based on the above literature survey the most common reasons for which AI methods, tools and techniques are applicable to SE are discussed:

- Automatic Programming (AP) in AI is synonymous with Software Engineering and this represents a new paradigm for SE in the future research.
- Expert systems technology is sufficiently successful and mature enough to provide significant solutions to certain aspects of the SE process and problem.
- AI development and maintenance environments are suitable for direct application to the SE process.
- AI methodology and techniques can be applied to the software design process.
- The AI rapid prototyping model is useful as a SE paradigm.
- Ai techniques reduces cost.
- Errors detected in coding will be isolated in the requirements stage.
- Changes need be made only at the requirements stage.

V. VARIOUS ARTIFICIAL INTELLIGENCE TECHNIQUES

The various Artificial Intelligence Technique used in the Software Development Process are listed below:

AI Technique	**Purpose**
Knowledge Based System	Used in the design phase of the software development process It manages the requirement phase , planning and project effort estimation
Neural Network	Eliminates the risk associated with modules in software maintenance and Used in the software engineering prediction outcomes.
Fuzzy logic	Reasoning the uncertainty
Genetic algorithm	Used in the software testing and generating test cases
Case Based Reasoning(CBR)	Used for finding out the duration or time taken to complete a project
Natural Language Processing	It helps in the user requirements and improves the phase of software development life cycle.
SBSE	Reformulating the software engineering problems as optimization problems
Rule induction	Used to defect prediction

Expert system	It uses the knowledge to overcome the risk management strategies during the software development process.
Genetic code	It develop automatically generate computer program and save the time in the coding phase.
Automated Tool	Use for system redesign. I t changes the traditional software development to expert system development
Automatic programming	Generation of program by computer usually based on specification
Simple decision making	Dealing with uncertainty
Intelligent Agent	It generate new intelligent software system for better communication
Simulated annealing& Tabu search	Used in the field of engineering
Probabilistic reasoning	Dealing with uncertainty

Table.1 Various Artificial Intelligence Techniques and its purposes

S.NO	AI LEARNING METHODS
1	Failure Driven Learning
2	Learning by being Told
3	Learning by Exploration

Table.2 Artificial Intelligence Learning methods

S.NO	APPROACHES	DESCRIPTION
1	classical approach	designing the AI
2	connectionist approach	letting AI develop

Table.3 Approaches of Artificial Intelligence

VI. CONCLUSION AND SUGGESTIONS:

In this paper the promising research work on applying AI techniques to solve some of the most important problems facing the software engineers is studied. The study suggests that there is now good progress in the use of AI techniques in SE. Furthermore, the development of new areas such as intelligent agents and their use in distributed computing, context aware and secure applications will require closer links between SE and AI in the future.

The important suggestions emerge from the above survey findings are Artificial Intelligence techniques applied to the software engineering process can have a major impact on reducing the time to market, cost of development and enhance the quality of software system and used to support the tasks of coding and testing. Artificial Intelligence techniques are well suited to the complex software engineering problems, because they are designed to deal with the most demanding challenges. Artificial Intelligence based systems with the help of automated tool or automated programming tool time can be saved in the software development process.

REFERENCES

[1] Jyoti tewari,Swati arya, Prem narayan singh ,"Approach of Intelligent Software Agents in Future Development",IJARCSSE, ISSN:2277128X , May 2013.
[2] Hany H Ammar, Walid Abdelmoez and Mohamed Salah Hamdi, "Software Engineering Using Artificial Intelligence Techniques: Current State and Open Problems" – ICCIT 2012.
[3] Mark Harman "The Role of Artificial Intelligence in Software Engineering" ACM computing surveys, 2011.
[4] Prince Jain,"Interaction between Software Engineering and Artificial Intelligence-A Review"-International Journal on Computer Science and Engineering (IJCSE), ISSN: 0975-3397, Vol 3 No.12 December 2011.
[5] Dr. Nachamai. M ,Senthil Vadivu and Tapaskar , "Enacted Software Development Process Based on Agent

methodologies",IJEST ,VOL.3, NO.11, November 2011.

[6] Farid Meziane, Sunil Vadera," Artificial Intelligence Applications for Improved Software Engineering Development: New Prospects" DOI: 10.4018/978-1-60566-758-4.ch014. 2010

[7] Farah Naaz Raza, "Artificial Intelligence Techniques in Software engineering-International multiconference of Engineers and computer scientists 2009 , ISBN 978-988.17012-2-0 Vol 1 – IMECS 2009.

[8] Parveen Ranjan Srivastava, and Tai-hoon Kim, Application of Genetic Algorithm in Software Testing, International Journal of Software Engineering and its Applications.Vol. 3, No.4, October 2009.

[9] Yang, H.-L., & Wang, C.-S. (2009). Recommender system for software project planning one application of revised CBR algorithm. ExpertSystems with Applications, 36(5), 8938–8945. doi:doi:10.1016/j.eswa.2008.11.050.

[10] Hooshyar, B., Tahmani, A., & Shenasa, M. (2008). A Genetic Algorithm to Time-Cost Trade off in project scheduling. In Proceedings of the IEEE World Congress on Computational Intelligence (pp. 3081-3086), Hong Kong. Washington DC: IEEE Computer Society.

[11] Zhen-Yu, Z., Wei-Yang, Y., & Qian-Lei, L. (2008). Applications of Fuzzy Critical Chain Method in Project Scheduling. In roceedings of the FourthInternational Conference on Natural Computation (pp. 473-477), Jinan, China. Washington DC: IEEE Computer Society.

[12] Bertolino, A. (2007). Software Testing Research: Achievements, Challenges, Dreams. In Proceedings of the IEEE International Conference on Software Engineering (pp. 85-103), Minneapolis, MN. Washington DC: IEEE Computer Society.

[13] Kobbacy, K. A., Vadera, S., & Rasmy, M. H. (2007). AI and OR in management of operations: history and trends. The Journal of the Operational Research Society, 58, 10–28. doi:10.1057/palgrave. jors.2602132

[14] Nand, S., Kaur, A., Jain S. (2007).Use Of Fuzzy Logic In Software Development. Issues in Information Systems. Volume VIII, No. 2, pp. 238-244

[15] Hu, Y., Chen, J., Rong, Z., Mei, L., & Xie, K. (2006). A Neural Networks Approach for Software Risk Analysis. In Proceedings of the Sixth IEEE International Conference on Data Mining Workshops (pp. 722-725), Hong Kong. Washington DC: IEEE Computer Society.

[16] Wappler, S., & Wegener, J. (2006). Evolutionary unit testing of object-oriented software using strongly-typed genetic programming. In Proceedings of the Eighth Annual Conference on Genetic and Evolutionary Computation (pp. 1925-1932), Seattle, WA. New York: ACM Press

[17] Yujia, G., & Chang, C. (2006). Capability-based Project Scheduling with Genetic Algorithms. In Proceedings of the International Conference on Intelligent Agents, Web Technologies and Internet Commerce (pp. 161-161), Sydney, Australia. ashington DC: IEEE Computer Society.

[18] Briand, L. C., Labiche, Y., & Shousha, M. (2005). Stress testing real-time systems with genetic algorithms. In Proceedings of the Conference on Genetic and Evolutionary Computation (pp. 1021- 1028). Washington DC. New York: ACM Press.

[19] Baresel, A., Binkley, D., Harman, M., & Korel, B. (2004). Evolutionary testing in the presence of loop-assigned flags: a testability transformation approach. In Proceedings of the ACM SIGSOFT International Symposium on Software Testing andAnalysis (pp. 108-118). Boston: ACM Press.

[20] Gupta, M., Bastani, F., Khan, L., & Yen, I.-L. (2004). Automated test data generation using MEA-graph planning. In Proceedings of the Sixteenth IEEE Conference on Tools with Artificial Intelligence (pp. 174-182). Washington, DC: IEEE Computer Society.

[21] Wallace, L., & Keil, M. (2004). Software project risks and their effect on outcomes. Communications of the ACM, 47(4), 68–73. doi:10.1145/975817.975819

[22] Baudry, B., Fleurey, F., Jezequel, J.-M., & Le Traon, Y. Automatic test case optimization using a bacteriological adaptation model: application to. NET components. In Proceedings of the Seventeenth IEEE International Conference on Automated Software Engineering (pp.253-256), Edinburgh, UK. Washington DC: IEEE Computer Society. (2002a).

[23] Baudry, B., Fleurey, F., Jezequel, J. M., & Le Traon, Y. (2002b). Genes and Bacteria for Automatic Test Cases Optimization in the. Net Environment. In Proceedings of the Thirteenth International Symposium on Software Reliability Engineering (ISSRE'02) 195- 206), Annapolis, MD. Washington DC: IEEE Computer Society.

[24] Briand, L. C., Feng, J., & Labiche, Y. (2002). Using genetic algorithms and coupling measures to devise optimal integration test orders. In Proceedings of the Fourteenth International Conference on Software Engineering and knowledge Engineering (pp. 43-50), Ischia, Italy. New York: ACM Press.

[25] Hindi, K. S., Hongbo, Y., & Fleszar, K. (2002). An evolutionary algorithm for resource-constrained project scheduling. IEEE Transactions on Evolutionary Computation, 6(5), 512–518.

[26] Jonathan Onowakpo Goddey Ebbah "Deploying Artificial Intelligence Techniques In Software Engineering, American Journal of Undergraduate Research, March 2002.

[27] Neumann, D. (2002). An Enhanced Neural Network Technique for Software Risk Analysis. IEEE Transactions on Software Engineering, 28(9),904–912. doi:10.1109/TSE.2002.1033229

[28] Shan, Y., McKay, R. I., Lokan, C. J., & Essam, D.L. (2002). Software project effort estimation using genetic programming. In Proceedings of the IEEE International Conference on Communications,Circuits and Systems (pp. 1108-1112), Arizona. Washington DC: IEEE Computer Society.

[29] Thwin, M. M. T., & Quah, T.-S. (2002). Application of neural network for predicting software development faults using object-oriented design metrics. In Proceedings of the Ninth International Conference on Neural Information Processing (pp.2312-2316), Singapore. Washington DC: IEEE Computer Society.

[30] Mark Harman, Bryan F. Jones, 'Search Based Software Engineering"- Elsevier science, 2001.

[31] T. Menzies, "Practical machine learning for software engineering and knowledge engineering," in Handbook of Software Engineering and Knowledge Engineering.World-Scientific, December 2001,

[32] Meziane, F., Vadera, S., Kobbacy, K., & Proudlove, N. (2000). Intelligent Systems in Manufacturing: Current Developments and Future Prospects. The International Journal of Manufacturing Technology Management, 11(4), 218–238.

[33] C. Ryan, Automatic re-engineering of software using genetic programming. Kluwer Academic Publishers, 2000.

[34] Ian Sommerville, Software Engineering (6th Edn.) (Addison Wesley Publishers, New York, New York, USA) 2000.

[35] Memon, A. M., Pollack, M. E., & Soffa, M. L. (1999). Using a Goal Driven Approach to Generate Test Cases for GUIs. In Proceedings of the Twenty-first International Conference on Software Engineering (pp. 257-266).

[36] J. Karlsson and K. Ryan, "A cost-value approach for prioritizing requirements," IEEE Software, vol. 14, no. 5, pp. 67–74, September/October 1997.

[37] Khoshgoftaar, T. M., & Lanning, D. L. (1995). A Neural Network Approach for Early Detection ofProgram Modules Having High Risk in the Maintenance Phase. Journal of Systems and Software,29, 85–91. doi:10.1016/0164-1212(94)00130-F

[38] Cheng, R., & Gen, M. (1994). Evolution program for resource constrained project scheduling problem. In Proceedings of the Proceedings of the 1st First IEEE Conference on Evolutionary Computation (pp. 736-741), Orlando, FL, USA.

[39] Stuart Russell and Peter Norvig, Artificial Intelligence: A Modern Approach (Prentice Hall Publishers, Upper Saddle River, New Jersey, USA) 1994.

[40] DeMillo, R.A., Offutt, A.J. (1991). Constraint-based automatic test data generation. IEEE Transactions on Software Engineering 17 (9), 900–910.

[41] Boardman, J. T., & Marshall, G. (1990). A knowledge- based architecture for project planning and control. In Proceedings of the UK Conference on IT (pp. 125-132), Southampton, UK. Washington DC: IEEE Computer Society.

[42] Seth Hock, Computers and Computing (Houghton Mifflin College Publishers, Boston, Massachusetts, USA) 1989.

[43] 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 (pp.1363-1368), Dayton, OH. Washington DC: IEEE Computer Society.

[44] 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.

[45] Roger S. Pressman,A Beginner's Guide (McGraw Hill Higher Education Publishers, New York, New York, USA) 1988.

[46] Booch, G. (1986). Object-Oriented Development. IEEE Transactions on Software Engineering, 12(2), 211–221.

[47] Sathi, A., Fox, M. S., & Greenberg, M. (1985). Representation of Activity Knowledge for Project Management. IEEE Transactions on Pattern Analysis and Machine Intelligence, PAMI-7(5),531–552. doi:10.1109/TPAMI.1985.4767701
[48] Wright, J. M., Fox, M. S., & Adam, D. (1984).SRL/2 User Manual: Robotic Institute, Carnegie- Mellon University, Pittsburgh, PA.
[49] Siemens, N. (1971). A Simple CPM Time-Cost Tradeoff Algorithm. Management Science, 17(6), 354–363. doi:10.1287/mnsc.17.6.B354