ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijircce.com

Vol. 5, Issue 4, April 2017

A Survey on the Role of Artificial Intelligence in Software Engineering

Tamalika Basu[1], Avantika Bhatia[1], Divya Joseph[1], Prof Ramanathan L[2]

B. Tech Student, School of Computer Science and Engineering (SCOPE), Vellore Institute of Technology, Vellore, Tamil Nadu, India[1]

Asst. Professor, School of Computer Science and Engineering (SCOPE), Vellore Institute of Technology, Vellore, Tamil Nadu, India[2]

ABSTRACT: Software is eating the world and automation is next in line. Software and Engineers have always wanted to automate everything. This paper validates the role of Artificial Intelligence (AI) in software engineering. Instead of humans manually creating all the software code and test automation, machines could aid in writing and executing test code, and continually improve as they learn from human input. This paper highlights and summarizes the various advantages and disadvantages that could be involved in mechanizing software engineering and discusses the role that agents could play in a system.

KEYWORDS: Artificial Intelligence, Software Engineering, automation, agents, computational intelligence

I. INTRODUCTION

Over the years, artificial intelligence has been subjected to myriad praise and criticism. It continues to evolve and be applied in almost every sphere of the technological world. Artificial intelligence involves developing systems capable of performing tasks of human cognition. Agents in this field are intelligent systems capable of performing activities such as decision-making and responding to their environment. In the recent years, researchers have begun to explore the role of intelligent agents could play in software development and engineering. A variety of reasons contribute to this study - the reluctance of software developers, inappropriateness of traditional software tools and ignorance of vital concerns in early stages of software development to mention a few. It is thus important to explore the integration of agent-based techniques in the field of software development. The aim in this paper is to analyze the views available regarding this integration and explore the possibilities of combining the roles of intelligent agents in the development of software systems.

II. LITERATURE REVIEW

Nicholas R Jennings [1] analyzes how agent based software contributes in solving real world problems. The paper argues that even though usual methodologies for software testing are fruitful in deriving conclusive results they have two main drawbacks. Firstly, the rigid interactions amongst the computational entities and secondly, the organizational structure of systems cannot be efficiently represented using the traditional methods. The proposal made in this paper centers around 'the adequacy hypothesis' which suggests that agent based approached can sufficiently enhance the ability to model, build and design distributed and complex systems. It also introduces 'the establishment hypothesis' which predicts that agent based techniques will be successful in the realm of software engineering. Presenting a qualitative approach, Jennings suggests that the techniques for dealing with complexity include decomposition; abstraction and organization are well managed by agent-based systems in the following manner. Agent based systems involve decomposition by partitioning the problem space in complex systems in terms of autonomous agents thus enabling high-level interactions and reducing the control complexity, which now lies in the individual components. Decision-making is resolved as each component makes a selection determined by the problem solver for each localized environment. This flexibility to take decisions at run-time about the nature and scope of functionalities simplifies

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijircce.com

Vol. 5, Issue 4, April 2017

complex systems. The abstractions of agent-based approach serve to solve problems in complex systems naturally and agents efficiently handle organizational relationships as they provide the flexibility to form groups in isolation, which are then added into the system in an incremental manner. The two disadvantages of agents presented are the unpredictable outcomes and patterns of interactions and the difficulty in predicting the complete behavior of the system on the basis of constituting components. Yet, agent based systems stand to increase throughput, efficiency and robustness of a system.

David Barstow [2] indicates that software engineering activities are knowledge intensive so the capability of agents to serve as knowledge representation systems and the ability to provide explicit representation of knowledge serve to improve software engineering costs. Knowledge in a system is utilized for decision-making, communication, inference and analysis. The artificial intelligence heuristic search paradigm, knowledge representation technique and rule based agents can be used to represent software engineering programming techniques, provide expertise, store and obtain knowledge about the application domain required during specification, requirement analysis and deployment as well as keep information regarding the history of the software by recording the motivations behind various decisions made during design, requirement analysis and implementation. The disadvantage of agent-based approach is their inabilities to alone solve completely the software engineering problems without the need for additional techniques to serve the purpose such as database techniques, communication techniques and user interface techniques. However, Barstow suggests that agent based techniques described above will be deemed necessary to solve software engineering issues and enhance productivity of a system provided the research and experiments problems are addressed effectively.

Witold [3] explores the role of computational intelligence in the field of software engineering and provides three regions where this intelligence can be applied. First, it is suggested that fuzzy models could be applied to software processes. It uses the concepts of neural networks and granular computing and indicates a high level of compatibility between software engineering paradigms and the foundation of granular computing. Second, it suggests that high dimensional software data can be visualized using self-organizing maps in neural networks. Third, it stresses on the ability to create logic models of software quality. It introduces the idea of search-based software engineering where in a metaheuristic searching algorithm could be used to find software solutions that are both feasible and possible. Witold goes on to express that computational intelligence has applications in a variety of aspects of software engineering, which involves testing, reliability and quality prediction, estimating cost, classifying software components and even creating models of software data.

Mark Harman [4] suggests that agent based algorithms can be employed effectively in software engineering issues with contributions in three main fields. Firstly, learning, classification and predictive methods help in the modeling and determination of software costs and model. This makes use of machine learning algorithms such as cased based reasoning, rule induction and the formulation of artificial neural networks to make project predictions and foresee defects. Secondly, effective optimization techniques and computational search methodologies can be integrated into a field call Search Based Software Engineering. This field designs software engineering issues as optimization issues, which are then, tackled using computational searching algorithms. The use of this type of engineering has a variety of applications in software testing, maintenance as well as requirements analysis and design. Thirdly, fuzzy and probabilistic works such as Bayesian scheme can be adapted to software engineering to determine software reliability and fits into the real world problems, such as analysis of software users, which are generally of fuzzy nature with incomplete and noisy data. Harman expands that artificial intelligence can be incorporated into software engineering to give insight to software engineers, search for software development strategies, exploit multicore computation, compile smart optimization into deployment software, to adapt software products and use cases to better suit agent oriented algorithms as well as obtain a harmonious trade off between human intervention and automation.

Ammar [5] recognized the application of artificial intelligence technique in the development process requirement engineering, architectural design, software coding and testing. In the segment of requirements engineering, artificial intelligence can be employed to clarify the natural language requirements, model software problems and develop ontological knowledge bases to handle requirements. Requirements can be prioritized and incomplete problems can be dealt with using intelligent computation techniques. The software issue of transformation of requirements into architectures can be targeted by developing service-oriented and product-line architectures that use AI techniques. In the coding phase, engineers can be assisted by using artificial intelligent techniques to automate programming or to guide the software developers in the programming process. Finally, testing case designs can be automated as well which reduces the work of software engineers. However, research still has to be carried out to find an optimal way for

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijircce.com

Vol. 5, Issue 4, April 2017

agent bases algorithms to optimally handle large pieces of dynamic codes and to give adequate guidance to any phase of the software development process.

F.Meziane [6] comments that in the current scenario of growing modernization and development in the smart-technical centre, artificial intelligence techniques like knowledge based systems, fuzzy logics and neural networks and data mining techniques have been adapted my several researchers and professionals to improve software development activities. Thus there is a noteworthy potential for using AI in almost all the phases in software development life cycle. The paper stated underlines the usage and vitality of Artificial Intelligence for Software Engineering and describes the important phases of software development and methods of AI, some being neural networks, genetic algorithms, ant colony optimization and some planning techniques. As per the survey conducted in this paper, the use of Genetic Algorithm is the most prevailing. Neural networks have been adapted for risk appraisal but Bayesian networks are more transparent and thus alluring to practice by the managers of the projects. Similar views were presented to the usage of Case based reasoning. The requirement and design phases have highlighted the importance to identify the errors that are faced in the initial stages during the development of a software. The survey has clearly suggested that though there is a remarkable progress in the applications of AI in the software engineering field yet there is a need of large scale evaluations and extensive researches to understand the coherence of different algorithms and approaches.

Vijila [7] in the paper states that development of a software is a laborious and a complex task but also essential. To programme in this field requires extensive and a wide variety of knowledge, both in the problem and the programming domain. This paper aspires to review the various algorithms and techniques developed in AI from the opinion of their application into the subject (Software Engineering). The paper agrees on the fact that automated engineering will have limitations and might be impractical in certain cases. The major problem being in the reconciliation of big programs. The paper emphasis on the fact that the basic concepts for the automated program's development should be language independent such as OOP (Object Oriented Programming), C++ ,etc.

Rech J [8] in his research paper manifests on the regulations that AI and Software Engineering share. Both have dealt with modeling around real time objects be it a business process or system model. The paper foregrounds individually on broader aspects of AI and SE and then relates them with each other pointing out that the intersections between the two may be rare now but are growing. Aspects of artificial intelligence AI researchers explain intelligence by being able to bring effects of intelligent behavior into use. AI has been classified under two goals- first being cognitive science and the second being engineering sciences.- aspects of software engineering :- the primary objective is to be able to support software engineers to develop a better model of the software which are faster with smarter tools and methods. Also to be able to construct software that could adapt to changing environment and can avert foreseeable obstacle and can rectify noticed defects. The article focuses on the researches done in the common ground connecting the two, forming the basis of knowledge based software engineering, computational intelligence and even ambient intelligence.

Wang [9] in his paper conveys in detail the two major constructs which are the software sciences and machine intelligence. Software science focuses on the structure of software systems as instructive and deportment information that can be personified and executed by generic computers to be able to create system behaviors as desired and machine intelligence. The paper signifies the disciplinary field of machine intelligence when the two constructs merge together and as a result produces fundamental theories, contemporary mathematics and engineering applications. The paper also proclaims the historic evolvement of the two along with the theoretical functions in an elaborated flow. The recent advances have also been depicted in an orderly manner. It shows that the investigation into machine-software intelligence service might lead to rudimentary findings towards the future generation and development.

R.Abreu [10] in his research paper intensifies on how the automated diagnosis of the faults in software can refine the productivity of the debugging process and is thus a vital technique for the software's development. The paper also talks about the various similarity coefficients, which are applied in the context of the software program to detect programming mistakes. The coefficients mentioned here are from various system diagnosis or automatic debugging tools. These coefficients are evaluated as per the Siemen Suite of benchmark faults. The experiments performed denote that Ochiai coefficient regularly outshines the coefficients which are currently used by the tools discussed (Pinpoint, Tarantula, AMPLE, etc). The paper talks about the amount of code that is needed to be inspected, this coefficient will improve 5% on an average over the next best proven technique and up to 30% in some specific cases.

Larkman [11] in his paper talks about importance of software testing and how failures can have catastrophic effects. Various areas of artificial intelligence involved in software testing have been highlighted which include white box,

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijircce.com

Vol. 5, Issue 4, April 2017

grey box and black box testing, however the author cities a lot of limitations of AI techniques including lack of empirical evidences and agree on the terms that they lack scientific basis. The author talks about framework for their proposed work and justifies decision support system through its objective being application of AI in software testing. The discussion takes up Fuzzy Cognitive Maps (FCM) and constructs a FCM with influences and temporal analysis using a test case. The study concludes with realizing that a lot of complex environment situations and important concepts cannot be covered in software testing which i totally agree on after taking a closer look on the work.

Kosko B. [12] talks about neural and fuzzy logic by giving a detailed perspective by incorporating applications along with theoretical knowledge. His work talks about the emerging fields of neural networks, which includes signal processing and image processing. His work also talk about hardware designs such as VLSI and optical. The detailed descriptions and solved theorem on Signal Analysis is notable.

Howe [13] in his paper mentions about automated test generations and how they have been playing a vital role in software testing ex: - Transaction based software's. The detailed work by the author on test generation process models is commendable. Further the author gives an insight on planning architecture where he highlights description of plan, creation of a plan, and its translations. Taking up various examples on test cases and satisfying the readers the author concludes that test cases should be flexible to accommodate extensive criteria's. The paper also talks extensively about various test case strategies keeping in mind valid and invalid test cases covering majority of the readers' perspective. However few questions such as the works application and its results on wider domains have not been covered, also while talking about testing goals no perspective regarding goal genesis has been provided.

Sorte [14] in his paper begins by stating commonalities in the worlds of AI and software engineering. The author makes lures the reader by highlighting the main contributions of Ai in Requirement specification by referring works of Yoichi Omori and Keijiro Araki. The author elucidates each stage of the life cycle starting from requirement tracing to software design and then covering the role of AI in code generation and finally software testing. Thus giving the reader enough proof to believe that AI has a major contribution throughout the entire software development process. The author concludes by referring to various expert systems, which are likely to play a major role in the software development life cycle, but AI will remain a significant common strength in varied domains.

C.Rajagopalan [15] explores the role of artificial intelligence in the field of non-destructive software testing. It suggests the use of artificial neural networks and knowledge based systems for the evaluation and testing of software systems. The benefits of fuzzy logic and neural networks have been proposed to be combined to gain maximum efficiency in non-destructive software testing. It highlights an architecture for intelligent systems for testing stating that they must be open-ended, allow incremental developments, incorporate various kinds of computing, use minimum memory, have modular design, require optimal software maintenance and call for rapid prototyping and testing. It is suggested that this architecture will allow for efficient testing capable of handling large types of problems.

III. CONCLUSION

Software engineering and artificial intelligence are two fields, that when combined attempt to solve problems and make easier the life of developers, testers and analyzers. This automated approach is a potential strategy that could be exploited to benefit aspects of software engineering and solve the problems faced by software engineers. In this paper, we surveyed promising applications of AI techniques to the realm of software engineering.

REFERENCES

1. Jennings, N.R., "On agent-based software engineering". Artificial intelligence, 117(2), pp.277-296, 2000.
2. Barstow, D., "Artificial intelligence and software engineering". 9th international conference on Software Engineering, pp. 200-211, 1987, March.
3. Pedrycz, Witold. "Computational intelligence as an emerging paradigm of software engineering.", 14th international conference on Software engineering and knowledge engineering, pp. 7-14, 2002.
4. Harman, Mark. "The role of artificial intelligence in software engineering.", First International Workshop on Realizing AI Synergies in Software Engineering, pp. 1-6, 2012.
5. Ammar, Hany H., Walid Abdelmoez, and Mohamed Salah Hamdi. "Software engineering using artificial intelligence techniques: Current state and open problems.", First Taibah University International Conference on Computing and Information Technology (ICCIT 2012), Al-Madinah Al-Munawwarah, Saudi Arabia, p. 52, 2012.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijircce.com

Vol. 5, Issue 4, April 2017

6. Meziane, Farid, and Sunil Vadera. "Artificial Intelligence in Software Engineering: Current Developments and Future Prospects." In Machine Learning: Concepts, Methodologies, Tools and Applications, pp. 1215-1236, 2012.
7. Vijila, J. Maria Merceline, and R. Abinaya. "Deploying artificial intelligence techniques in software engineering."
8. Rech, J. and Althoff, K.D., "Artificial intelligence and software engineering: Status and future trends.", KI, 18(3), pp.5-11, 2004.
9. Wang, Y., "Convergence of Software Science and Computational Intelligence: A New Transdisciplinary research Field", In Software and Intelligent Sciences: New Transdisciplinary Findings, pp. 1-13, 2012.
10. Abreu, R., Zoeteweij, P., & Van Gemund, A. J., "An evaluation of similarity coefficients for software fault localization". In Dependable Computing, 2006. PRDC'06. 12th Pacific Rim International Symposium, pp. 39-46, 2006.
11. Larkman, D., Mohammadian, M., Balachandran, B., & Jentzsch, R., "Fuzzy cognitive map for software testing using artificial intelligence techniques.", in IFIP International Conference on Artificial Intelligence Applications and Innovations, pp. 328-335, October 2010.
12. Kosko, Bart. "Neural networks and fuzzy systems: a dynamical systems approach to machine intelligence/book and disk." Vol. 1Prentice hall, 1992.
13. Howe, A. E., Von Mayrhauser, A., & Mraz, R. T., "Test case generation as an AI planning problem", in Knowledge-Based Software Engineering (pp. 77-106). Springer US, 1997.
14. Sorte, Bhagyashree W., Pooja P. Joshi, and Vandana Jagtap. "Use of Artificial Intelligence in Software Development Life Cycle: A state of the Art Review."
15. Rajagopalan, C., Raj, B. and Kalyanasundaram, P.,"The role of artificial intelligence in non-destructive testing and evaluation", Insight, 38(2), pp.118-23, 1996.