www.ijecs.in
International Journal Of Engineering And Computer Science ISSN:2319-7242
Volume 3 Issue 4 April, 2014 Page No. 5288-5297

A Review of Research Work in Software Engineering

Madhu Kumari[1], Meera Sharma[2], Ajay Kumar[3]

[1]Delhi College of Arts & Commerce
University of Delhi, India
mesra.madhu@gmail.com

[2]Swami Shraddhanand College
University of Delhi, India
meerakaushik@gmail.com

[3]Shivaji College
University of Delhi, India
Ajaykr.bhu@gmail.com
Corresponding Author

Abstract: The development of open source software is a multidisciplinary approach and it requires different areas of expertise, knowledge, tools and techniques. The open source software development has an important role. During the last decades, open source software development has changed the dynamics of software engineering research and added different new domains. In closed source software, the data related to software development, bug reported before release and post release was not available. It was difficult for researchers to validate their methods and models due to non-availability of data. In open source software, different repositories namely Source control repositories, Bug repositories, Archived communications, Deployment logs, Code repositories are available. Researchers are developing methods to mine useful information from these repositories to improve the quality of software projects. Different machine learning techniques have been applied to determine the level of severity and priority of bugs, to find the buggy module, security bugs and right developers. In this paper we are trying to focus on various domains such as Artificial Intelligence based Software Engineering to develop new tools, Model Based Software Engineering, Search Based Software Engineering , Role of Software Engineering in Cloud Computing, Quantitative and Qualitative Software Engineering, Empirical Software Engineering ,regression based prediction models and machine learning techniques used to predict the bug fix time, man power involved in fixing that bug , assign a bug to the right fixer and .

Keywords: Software engineering , Artificial Intelligence, Search based software engineering, Model based software engineering, Machine learning techniques, quantitative and qualitative software engineering, Empirical software engineering **.**

1. Introduction

As pointed out by David Rice, "Like cement, software is everywhere in modern civilization. Software is in your mobile phone, in your home computer, in cars, airplanes, hospitals, businesses, public utilities, financial systems, and national defense systems. "[1]

Software is basically computer instructions or data used in software engineering which refers to the process of developing software initially, then repeatedly updating it for various reasons for development**.** Software engineering (SE) is concerned with the quantifiable approach to the development, design, operation, and maintenance of software.

For the last 50 years, many critical issues (low quality and productivity, and high cost and risk, unreliable, un-maintainable, inconsistent etc.) have existed in the old-established software development process. The major reason behind this, unorganized, nonlinear and not well engineered structure of software engineering paradigm in which a small change at one place in a nonlinear system may bring a large variation to the entire system in a later state– the **"Butterfly-Effec**t". But in the recent era the improved software development methodology gives the better understanding of complex system in terms of their components, so that all the software engineering research activities are performed linearly, partially, locally, quantitatively and qualitatively.

The rest of the paper is organized as follows. Section 2 of the paper presents an overview of the current trends in software engineering that includes Artificial Intelligence techniques can be used in the field of Software Engineering to develop new tools, Model Based Software Engineering, Search Based Software Engineering, Role of Software Engineering in Cloud Computing, Quantitative and Qualitative Software Engineering, Empirical Software Engineering and the usage of Machine Learning Techniques. Description of Machine Learning techniques have been presented in section 3. Section 4 briefly motivates on the regression models and finally the paper is concluded in section 5.

2. Current Trends in Software Engineering

It was only in the last decade of development that the production of higher quality and lower cost software started to emerge and **software engineering** gained its present importance. Today, software represents the key to success of many computer systems, and the factor of differentiation of organizations that have it. Software has become an essential component in business decisions and a basis in scientific research and engineering problem solving. It also represents a significant component in industrial, transportation, medical, telecommunications, military and many other types of systems. In the modern world, the software is virtually inevitable and ubiquitous. On beginning of the 21st century, understanding of the software has changed and is accepted by the public as technological reality in the future development. Present software development is similar to that in the early days. Although programming tools and languages are more advanced, technical equipment and information easily available, and the possibilities wider, these factors do not apply to specific development methodologies which remain equivalent to the first basic models.

Nowadays, the competitive advantages of developing software products derive mainly from speed and flexibility. To be competitive, organizations need to be closely associated with their customers, and be open to changes in user requirements. For this reason, today, the traditional models of development, with organizations that are willing to innovate, are being replaced by agile methods. All these "new" models of development have their own advantages and disadvantages – there is no best method. Selection of the most appropriate methodology will always depend on the organization, the scope and complexity of projects, cooperation of clients, staff experience, and other external and internal factors.

Software Engineering can be used to make the software simple and cost efficient. For that purpose new areas are explored to develop better techniques and tools to manage the complexity of software systems. In this paper, we have discussed different trends of software engineering.

2.1 Artificial Intelligence and software Engineering

AI is a branch of computer science which is concerned with the study and modeling human cognition using intelligence computer system . According to the Mark Harman[]Artificial Intelligence is about making machines intelligent, while software engineering is the activity of defining, designing and deploying some of the most complex and challenging systems mankind has ever sought to engineer. Though software engineering is one of the most challenging of all engineering disciplines, it is often not recognized as such, because software is so well concealed.

There are various Artificial intelligences techniques such as knowledge based systems, neural networks, fuzzy logic, data mining, natural language processing techniques, neural networks, genetic algorithms, fuzzy logic and ant colony optimization which have potential to improve the theory and the implementation of modeling and designing the software development paradigm.

In traditional software engineering development process begins at the requirement analysis and specification phase and end at the testing phase. At each of these phases different types of knowledge are required (design knowledge, domain knowledge and programming knowledge).There are various error can occur at any phase of software development process. Such errors are usually difficult and expensive to correct [62].

The basic problem of software engineering is the long gap between requirement specification and the delivery of the product. This long gap must be reduced before product arrival. In addition, there is the problem of phase independence of requirements, design and codes. Phase independence means that any decision made at one level becomes fixed for the next level. Thus, the coding team is forced to recode whenever there is change in design [61].

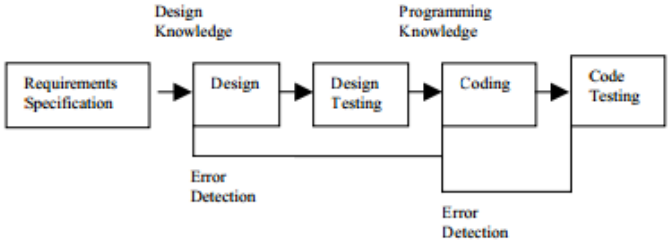

Figure 1.Traditional software development process[61]

Knowledge-based techniques in AI can be used to modify this traditional approach the AI technique that handles this problem is automated programming which results in reusable code[63][64]. 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. This technique requires constraint propagation technique. With the help of automated programming approach AI based systems are free from risk management strategies [61].

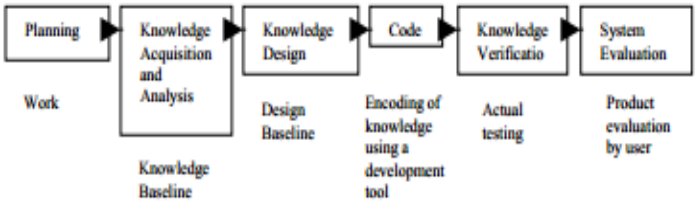

Figure2 : Expert System development process

Table1: Research Works in Software Engineering based on AI

Authors [Ref]	Objective Of Research
[58] Ammar, Abdelmoez and Hamdi	This paper surveys the application of artificial intelligence approaches to the software engineering processes. These approaches can have a major impact on reducing the time to market and improving the quality of software systems. They surveyed research in the development activities of requirements engineering, software architecture design, and coding and testing processes.
[59] Ebbah	The author focuses on techniques developed (or that are being developed) in artificial intelligence that can be used to solve the problems associated with software engineering

	processes.
[60] Prince Jain	They describe the interaction area between AI and SE. But before interaction, there is need of proper knowledge, discussion and understanding of factors and issues that originates while performing interaction. There is needed to be reducing and resolving the factor and issues which comes between while communicating the AI and SE
[61] Raza	The author highlights a comparative study between the traditional software development process and expert system development process. This paper also highlights how AI can be used to modify this traditional approach .With the help of automated programming structure AI based systems are free from risk management strategies. Because of Artificial Intelligence Techniques in Software Engineering (AITSE) we can reduce the development time in software development. Coding phase in software development process can be changed into Genetic Code.

2.2 Model Based Software Engineerin g

"Test data creation is a tedious, expensive, time-consuming and error-prone activity. Model driven approaches are springing up to aid up auto generation of test data."

Anjaneyulu Pasala PhD
Senior Research Scientist
Infosys Labs, Infosys Ltd.

Model based software engineering is a software development methodology which is used to create and exploit model in such a way that is easily understandable. It gives the abstract representation of the knowledge rather than the computing concepts. Its main objective is to change the software development process from manual coding to automatic code generation to achieve code reuse and perform maintenance and product development through the use of software modeling technology.

Table 2: Research Works in Software Engineering based on MBSE

Authors [Ref]	Objective Of Research
[6] Basha, Moiz And Rizwanullah	The authors presents the state-of-the-art of the Model-Based Software Development, the Model-Based Software Engineering (MBSE), Model Centric Software Development (MCSD) and Domain Engineering process with the specific domain .They demonstrates the purpose of DARE-COTS tool along with the scope of product lines. They also highlights the research challenges in terms of Multi Aspect Modeling.
[7] Karsai, JSztipanovits, Ledeczi, and Ted Bapty	The authors describe a model-integrated approach for embedded software development that is based on domain-specific, multiple view models used in all phases of the development process. They also demonstrate the principles and techniques of model-integrated embedded software development, as well as the capabilities of the tools supporting the process.

2.3 Search Based Software Engineering

In the past five years, Search Based Software Engineering (SBSE) becomes an emerging trend in software engineering research problem. It deals with metaheuristic search such as genetic algorithms (GA), simulated annealing and tabu search. It is based on computational search and optimization technique that is widely applicable to almost all phases of the software development process. SBSE has been applied throughout the software engineering lifecycle such as testing, debugging and maintenance, design, management, verification, general aspects, requirement engineering, project planning and re-engineering.

The term SBSE was first used by Harman and Jones [18] in 2001. This was the first time in the literature that it was suggested that search based optimization could be applicable right across the full spectrum of activities in Software Engineering.

Fig. 3 and 4 show the trend of publications on SBSE and Software Engineering topic area [17].

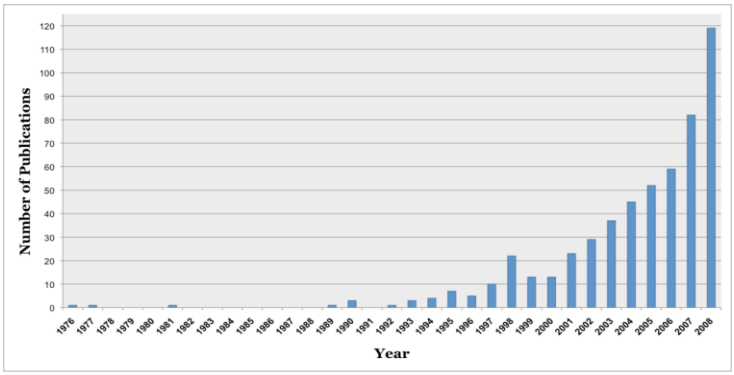

Figure 3 : The trend of publications on SBSE.

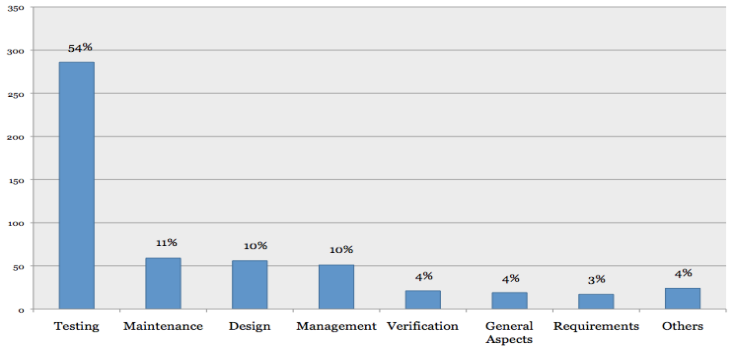

Figure 4: The trend of publications on Software Engineering Topic Areas

Table 3: Research Works in Software Engineering based on SBSE

Authors [Ref]	Objective Of Research

[19] Alshahwan and Harman [20] Fraser and Arcuri [21] Jia and Harman [22] Lakhotia, Harman, and Gross	SBSE applications including tools for testing.
[23] Mitchell and Mancoridis	Modularization
[24] Goues, Nguyen, Forrest, and Weimer	Bug Fixing
[25] Antoniol, Penta, and Harman	Authors show that an ordering-based genome encoding (with tailored cross over operator) and the genetic algorithm appear to provide the most robust solution, though the hill climbing approach also performs well. The best search technique results reduce the project duration by as much as 50%.
[26] Davis	Project Scheduling
[27] Falkenauer	Grouping Problems
[28] Bagnall, Rayward-Smith, and Whittley	The life-cycle of requirements engineering
[29] Aguilar-Ruiz, Ramos, Riquelme, and Toro [30] Antoniol, Penta, and Harman [31] Antoniol, Penta, and Harman [32] Burgess and Lefley [33] Dolado [34] Kirsopp, Shepperd, and Hart	Project planning and cost estimation .
[35] Baresel, Binkley, Harman, and Korel [36] Baresel, Sthamer, and Schmidt [37] Bottaci [38] Briand, Feng, and Labiche [39] Briand, Labiche, and Shousha [40] Guo, Hierons, Harman, and Derderian [41] Harman, Hu, Hierons, Wegener, Sthamer, A. [42] Li, Harman, and Hierons [43] McMinn, Harman, Binkley, and Tonella [44] J. Wegener, A. Baresel, and H. Sthamer	Testing
[45] Bouktif, Antoniol, Merlo, and Neteler [46] Fatiregun, Harman, and Hierons. [47] Harman, Hierons, and Proctor [48] Mitchell and Mancoridis [49] Mitchell and Mancoridis. [50] O'Keeffe and O'Cinneide. [51] Seng, Bauer, Biehl, and Pache. [52] O. Seng, J. Stammel, and D. Burkhart	Automated Maintenance
[53] Canfora, Penta, Esposito, and Villani	Service oriented software engineering
[54]Cohen, Kooi, and Srisa-an [55] Cooper, Schielke, and Subramanian	Compiler optimization
[56] Bouktif, Sahraoui, and Antoniol. [57] Khoshgoftaar, L. Yi, and Seliya	Quality assessment

2.4 Cloud Computing and Software Engineering

Cloud computing enables on-demand network access to shared pool of computing resource such as storage, software's, different platforms etc. Cloud computing lets a consumer to consume it's services and are charged according to the service that they have used. When engineering is disciplines to cloud computing it forms cloud engineering. Cloud engineering invests the methods and tools of software engineering in conceiving, developing, operating and maintaining cloud computing systems and its services.

By combining cloud computing and software engineering, we can meet the individually challenges faced by the cloud computing and software engineering. For example, a major challenge in software engineering is to manage the runtime QoS of loosely coupled service in distributed environment. Cloud computing can meet this challenge through resource allocation and virtualization.

On the other hand, cloud computing struggles both with providing interoperability across different clouds and with the rapid development and adaptation to, ever-changing business environments and requirements. SOA's standard interfaces and protocols could help address this interoperability challenge [http://www.infoq.com/articles/ieee-software-engineeering-services-cloud-computing/].

Table 4: Research Works in Software Engineering based on SBSE

Authors [Ref]	Objective Of Research
[66] Yadav, Khatri and Singh	They proposed a framework that makes cloud smarter and intelligent than simple cloud by using Artificial intelligence techniques. AI based cloud service models for higher education will help in cutting the costs spent on buying resources.

	With the help of this model quality of cloud will increase and make our education system more modern and transparent with smaller budgets.
[67] Yadav, Singh and Kumari	They concluded that software with minimum bug complexity and minimum operating time are much reliable. These software follows delayed and more delayed exponential distribution while newer software component follows exponential distribution that are more prone to failure's and less reliable. Their proposed model is based on Weibull distribution for hardware and delayed exponential for software.
[68] Thanakornworakij, Nassar, Leangsuksun1, Păun	They have developed a reliability model and estimate the mean time to failure and failure rate based on a system of k nodes and s virtual machines under four major scenarios that are combinations of software and hardware failure correlation. They demonstrate that if the failure of the hardware and/or the software in the system have a degree of dependency, the system becomes less reliable, which means that the failure rate of the system increases and the mean time to failure decreases. Additionally, an increase in the number of nodes decreases the reliability of the system.
[70] Dai, Yang, Dongarra and Zhang	They conducted a systematic approach on reliability modeling and analysis of cloud service They elaborated various types of possible failures in a cloud service such as overflow failure, timeout failure, resource missing failure, network failure, hardware failure, software failure, and database failure. Based on these failures they have developed holistic reliability model using Markov models, Queuing Theory and Graph Theory.A new algorithm is proposed to evaluate cloud service reliability based on the developed model by using Bayesian approaches and Graph Theory.

2.5 Quanlitative and Quantitative Software Engineering

Software engineering commonly classified as either quantitative or qualitative. But these two distinct research methods exist simultaneously in software engineering.

Qualitative Software engineering deals with the data that is represented as words and pictures not numbers. Basically the qualitative software engineering is the study of the complexities of human behavior (e.g. motivation, communication, and understanding). In qualitative research method data are descriptive, expressed in terms of words like experiments, questionnaires and psychometric tests rather than numbers. Qualitative data is sometimes referred to as categorical data. They need to be quantifying to do analysis. But in quantitative approach, data are anything that can be expressed as a number, or quantified. Both types of data are valid types of measurement, and both are used in software engineering methodology but only quantitative data can be analyzed statistically because of its numeric form, and thus more rigorous assessments of the data are possible.

Table 5: Characteristics of quantitative and qualitative research [15].

Quantitative	Qualitative
Objective	Subjective
Research questions: How many? Strength of association?	Research questions: What? Why?
"Hard" science	"Soft" science
Literature review must be done early in study	Literature review may be done as study progresses or afterwards
Test theory	Develops theory
One reality: focus is concise and narrow	Multiple realities: focus is complex and broad
Facts are value-free and unbiased	Facts are value-laden and biased
Reduction, control, precision	Discovery, description, understanding, shared Interpretation
Measurable	Interpretive
Mechanistic: parts equal the whole	Organismic: whole is greater than the parts
Report statistical analysis. Basic element of analysis is numbers	Report rich narrative, individual; interpretation. Basic element of analysis is words/ideas.
Researcher is separate	Researcher is part of process
Subjects	Participants
Context free	Context dependent
Hypothesis	Research questions
Reasoning is logistic and deductive	Reasoning is dialectic and inductive
Establishes relationships, causation	Describes meaning, discovery
Uses instruments	Uses communications and observation
Strives for generalization Generalizations leading to prediction, explanation, and understanding	Strives for uniqueness Patterns and theories developed for understanding
Highly controlled setting: experimental setting (outcome oriented)	Flexible approach: natural setting (process oriented)
Sample size: n	Sample size is not a concern; seeks "informal rich" Sample
"Counts the beans"	Provides information as to "which beans are worth counting"

2.6 Empirical Software Engineering

Empirical software engineering requires the scientific use of quantitative and qualitative data to understand and improve the software product, software development process and software management.

Empirical software engineering do not only deal with the software development process but also deal with the people involved in software development. So, the integration of both quantitative and qualitative research methods are necessary in software engineering development process. In the early days, empirical research in software engineering has been based on quantitative approach only but now these days studies in empirical software engineering are often involve software developers as well as software development organization.

Table 6: Research Works in Software Engineering based on Empirical Software Engineering

Authors [Ref]	Objective Of Research
[14] Lázaro, and Marcos	The authors observed that if these two SE research methods are applied separately the results obtained are incomplete. Hence, it is difficult to choose definitively between quantitative and qualitative methods when embarking on a specific research. To address this problem, a new research method based on integrating quantitative and qualitative methods is proposed and here a first approach to a new research method is proposed that is similar to the implementation of integrated qualitative and quantitative methods in the social sciences. Specifically, of the three types of integration taken from the field of social sciences, complementation is chosen, and this modified and redefined for improved usage in the field of SE.

3. Machine Learning Techniques and Software Engineering

In the traditional approach, usage of statistical estimation models provides incomplete, unreliable and poor performance result. They do not deal with categorical data, missing data points, spread of data points, and data with outliers. Machine Learning techniques remove all the demerits of traditional approach by improving performance through mechanizing the acquisition of knowledge from experience. ML algorithms not only can be used to build tools for software design, development and maintenance task but also can be used in tackling software engineering problem.

Machine learning deals with the issue of how to build computer programs that improve their performance at some task through experience [12]. Machine learning algorithms have been utilized in: (1) data mining problems where large databases may contain valuable implicit regularities that can be discovered automatically; (2) poorly understood domains where humans might not have the knowledge needed to develop effective algorithms; and (3) domains where programs must dynamically adapt to changing conditions [12].

Machine learning also identifies the software development and maintenance tasks in the following areas : requirement engineering (knowledge elicitation, prototyping); software reuse (application generators); testing and validation; maintenance (software understanding); project management (cost, effort, or defect prediction or estimation). Machine learning is the subfield artificial intelligence that allow computers to improve their performance and automatically produce (induce) models, such as rules and patterns, from data. Hence, machine learning is closely related to fields such as data mining, statistics, inductive reasoning, pattern recognition, and theoretical computer science.

Table 7 contains a list of software engineering tasks for which ML methods are applicable [13].

Table 7: SE tasks and applicable ML methods.

SE tasks	Applicable type(s) of learning
Requirement engineering	AL, BBN, LL, DT, ILP
Rapid prototyping	GP
Component reuse	IBL (CBR[4])
Cost/effort prediction	IBL (CBR), DT, BBN, ANN
Defect prediction	BBN
Test oracle generation	AL (EBL[5])
Test data adequacy	CL
Validation	AL
Reverse engineering	CL

The field of Machine Learning includes: supervised learning, unsupervised learning and reinforcement learning.

- **Supervised learning :**

Supervised learning includes neural networks (NN), Bayesian learning (BL), Case-based reasoning(CBR), Decision trees(DT), Instance-based learning (IBL), Support vector machines(SVM), Regression analysis, Information fuzzy networks (IFN), inductive logic programming (ILP), concept learning (CL), genetic algorithms (GA) and genetic programming (GP), analytical learning (AL), combined inductive and analytical learning (IAL), ensemble learning (EL), explanation-based learning(EBL) etc.

- **Unsupervised learning :**

Unsupervised learning includes Artificial neural network ,Data clustering, Expectation-maximization algorithm, Self-organizing map, Radial basis function network,Vector Quantization, Generative topographic map, Information bottleneck method etc.

- **Reinforcement learning :**

Reinforcement learning includes reinforcement learning (RL),Temporal difference learning, Q-learning, Learning Automata, Monte Carlo Method ,SARSA etc.Table 2 demonstrates the distribution of ML algorithms in the seven SE application areas.

Table 8: ML methods in SE application areas.

	NN	IBL CBR	DT	GA	GP	ILP	EBL	CL	BL	AL	IAL	RL	EL	SVM
Prediction	√	√	√	√	√	√		√	√					
Discovery	√					√	√		√					√
Transformation	√	√		√	√									
Generation		√	√	√	√	√	√	√		√				
Reuse		√	√	√			√							
Acquisition	√		√			√	√	√						
Management		√												

Many machine learning techniques like Artificial Neural Networks, Decision Tree, Linear Regression, Support Vector Machine, Fuzzy Logic, Genetic Algorithm, Empirical Techniques, and Theory based techniques has been used by researchers to tackle software engineering problems. In the 21st century, using the technique is not a big deal , the problem is ,which technique is most appropriate for a particular data set.
There are various ML tools for machine learning and statistical analysis including SAS, SPSS, Weak and the R language – allow deep analysis of smaller data and Mahout, Pentaho or Rapid Miner – allow shallow analysis of big-data and tools such as Spark, Twister, HaLoop, Hama and Graph Lab – facilitate deeper analysis of big-data .

The information given in figure 5 indicates that the increased interest in Machine Learning techniques in software engineering.

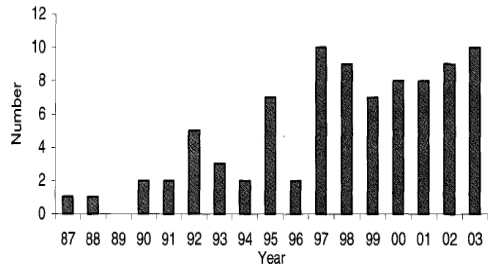

Figure 5: Publications on applying ML algorithms in SE.

4. **Regression Model**

In Quantitative Software Engineering, research methods work with data in numerical form collected from representative open source repositories and analyzed through statistical methods. One of the the statistical method is regression. The objective of regression analysis is to identify the relationship between the dependent and independent variables, eliminating insufficient and imprecise variables, and reduce the complexity of the problem so that the research reach their goals or conclude some results.

Authors [Ref]	Objective Of Research
[2] Sharma, Kumari and Singh	They Proposed prediction models based on linear regression to predict the bug attributes and to determine their linear relationships
[3] Chaturvedi and Singh	An attempt to demonstrate the applicability of machine learning algorithms to predict the bug severity level of summary of bug reports of NASA project.
[4] Bhattacharya and Neamtiu	They demonstrate that, the bug-fix time in open source projects is not influenced by the bug-opener's reputation. They proposed that various bug report attributes which have been previously used to build bug-fix time prediction models do not always correlate with bug-fix time.
[5] Yan et al	It is clear from the literature that very few efforts have been made to predict the severity of a bug and bug fix time by using the other bug attributes. Sharma et al.
[8] Sharma, Bedi, Chaturvedi and Singh.	They have shown the applicability of machine learning algorithms namely to predict the bug priority. They reported that cross-project priority prediction worked with 72% accuracy.
[9] Currently, Tian et al.	They proposed a new approach to predict severity of a bug automatically in particular BM25-based document similarity function. They focused on predicting fine-grained severity labels, namely the different severity labels of Bugzilla . They proposed a new approach ,automatically analyzes bug reports reported in the past along with their assigned severity labels, and recommends severity labels to newly reported bug reports .
[10] Bhattacharya and Neamtiu	They proposed an idea of reducing tossing path lengths of a bug to 1.5-2 tosses for most bugs, which represents a reduction of up to 86.31% compared to original tossing paths. This reduction in tossing path length improved triaging accuracy and got 83.62% prediction accuracy in bug triaging.They have shown how intra-fold updates are beneficial for achieving higher prediction accuracy in bug triaging when using classifiers in isolation.
[11] Kim and Whitehead	They demonstrates the distribution of bug counts for each bug fix time .

5. Conclusion

Software development is a very complicated process and it needs to be improved day by day .Now these days, Software Programmer has a different view in the field of Software Engineering development paradigm. Software engineering term first appeared in the 1968 NATO Software Engineering . During past 45 years, an exponential growth has been occurred in the several fields of software engineering such as process models, the software development methods, the test paradigm, the quality assurance paradigm, the documentation paradigm, the maintenance paradigm, the project management paradigm, etc.

During the last decades, many critical issues (low quality and productivity, and high cost and risk, unreliable, un-maintainable, inconsistent etc.) have existed in the old-established software engineering development process.
Software Engineering can be used to make the software simple and cost efficient. For that purpose new areas are explored to develop better techniques and tools to manage the complexity of software systems. In this paper we have demonstrate an abstract view of different research area associated with software development paradigm. We have discussed different trends of software engineering such as Artificial Intelligence techniques can be used in the field of Software Engineering to develop new tools, Model Based Software Engineering, Search Based Software Engineering, Role of Software Engineering in Cloud Computing, Quantitative and Qualitative Software Engineering, Empirical Software Engineering and the usage of Machine Learning Techniques in Software Engineering.

References

[1] Rice D (2008) Geekonomics: the real cost of \ insecure software. Addison-Wesley, Upper Saddle River.

[2] Sharma Meera, Kumari Madhu, and Singh VB, "Understanding the Meaning of Bug Attributes and Prediction Models" I CARE '13 Proceedings of the 5th IBM Collaborative Academia Research Exchange Workshop Article No. 15 ,ACM New York, NY, USA ©2013

[3] Chaturvedi K.K. and Singh V.B., 2012. Determining bug severity using machine learning techniques. In Proceedings of International Conference Software Engineering (CONSEG). CSI-IEEE, 2012, 378-387. DOI= http://ieeexplore.ieee.org/ 10.1109/CONSEG.2012.6349519.

[4] Bhattacharya, P. and Neamtiu, I. 2010. Bug-fix Time Prediction Models: Can We Do Better? In Proceedings of the 8th Working Conference on Mining Software Repositories (New York, NY, USA, 2012). ACM, 207-210. DOI= http://dl.acm.org/10.1145/1985441.1985472.

[5] Yan, Z., Chen, X., and Guo, P. 2010. Software Defect Prediction Using Fuzzy Support Vector Regression. In Proceedings of the 7th International Symposium on Neural Networks (Shanghai, China, June 6-9, 2010). Springer-Verlag Berlin Heidelberg 2010, 17-24. DOI= http://link.springer.com/chapter/10.1007%2F978-3-642-13318-3_3.

[6] Basha, N Md Jubair ., Moiz , Salman Abdul ., and Rizwanullah, Mohammed., Model Based Software Development: Issues & Challenges. Special Issue of International Journal of Computer Science & Informatics **(**IJCSI), ISSN **(**PRINT**)** : 2231–5292, Vol.- II, Issue-1, 2,226-230.

[7] Karsai, Gabor., Sztipanovits, Janos., Ledeczi, Akos., and Bapty Ted .Model-Integrated Development of Embedded Software. PROCEEDINGS OF THE IEEE, VOL. 91, NO. 1, JANUARY 2003

[8] Sharma,M., Bedi,P., Chaturvedi, K.K., and Singh., V.B. 2012. Predicting the Priority of a Reported Bug using Machine Learning Techniques and Cross Project Validation. In Proceedings of the 12th International Conference on Intelligent Systems Design and Applications (Kochi, India, Nov. 27-29, 2012).IEEE, 539-545.DOI=http://ieeexplore.ieee.org/ 10.1109/ISDA.2012.6416595.

[9] Tian, Y., David L., and Sun, C. 2012. Information Retrieval Based Nearest Neighbor Classification for Fine-Grained Bug Severity Prediction. In Proceedings of the 19th Working Conference on Reverse Engineering (WCRE),(15-18 Oct. 2012). 215-224.

[10] Bhattacharya, P. and Neamtiu, I. 2010. Fine-grained incremental learning and multi-feature tossing graphs to improve bug triaging. In Proceedings of the International Conference on Software Management(Washington, DC, USA, 2010).ACM, 1-10. DOI= http://dl.acm.org/10.1109/ICSM.2010.5609736.

[11] S. Kim and E. J. Whitehead, Jr. How long did it take to fix bugs? In MSR, 2006.12. T. Mitchell, Machine Learning, McGraw-Hill, 1997.

[12] Zhang. Du and Tsai, J.P. Jeffery Machine Learning and Software Engineering , Software Quality Journal, 11, 87–119, 2003

[13] Du Zhang, Applying Machine Learning Algorithms In Software Development, Department of Computer Science, California State University, Sacramento, CA 95819-6021, zhangd@ecs.csus.edu

[14] María Lázaro, Esperanza Marcos: An Approach to the Integration of Qualitative and Quantitative Research Methods in Software Engineering Research. PhiSE 2006

[15] Imperial COE, 2006. John D. Anderson, Superintendent of Schools. Page 3. Qualitative and Quantitative Research.

[16] Harman, M. 2006. Search Based Software Engineering, In Workshop on Computational Science in Software Engineering.

[17] Harman Mark, Mansouri S. Afshin, Zhang Yuanyuan, Search Based Software Engineering: Trends, Techniques and Applications. ACM Computing Surveys, Volume 45 Issue 1, November 2012, Article No. 11

[18] Harman, M. and Jones, B. F. (2001a). Search-based Software Engineering. Information & Software Technology,43(14), 833–839.

[19] N. Alshahwan and M. Harman, "Automated web application testing using search based software engineering," in 26thIEEE/ACM International Conference on Automated Software Engineering (ASE 2011), Lawrence, Kansas, USA, 6th - 10th November 2011, pp. 3 – 12.

[20] G. Fraser and A. Arcuri, "Evosuite: automatic test suite generation for object-oriented software," in 8th European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering (ESEC/FSE'11). ACM, September 5th - 9th 2011, pp. 416–419.

[21] Y. Jia and M. Harman, "Milu: A customizable, runtime-optimized higher order mutation testing tool for the full CLanguage," in 3rd Testing Academia and Industry Conference -Practice and Research Techniques (TAIC PART'08),Windsor, UK, August 2008, pp. 94–98.

[22] K. Lakhotia, M. Harman, and H. Gross, "AUSTIN: A tool for search based software testing for the C language and its evaluation on deployed automotive systems," in 2nd International Symposium on Search Based Software Engineering (SSBSE 2010), Benevento, Italy, September 2010, pp. 101 – 110.

[23] B. S. Mitchell and S. Mancoridis, "On the automatic modularization of software systems using the bunch tool,"IEEE Transactions on Software Engineering, vol.

32, no. 3,pp. 193–208, 2006.
[24] C. Le Goues, T. Nguyen, S. Forrest, and W. Weimer, "GenProg: A generic method for automatic software repair,"IEEE Transactions on Software Engineering, vol. 38, no. 1,pp. 54–72, 2012.
[25] Antoniol, Giulio., Penta, Massimiliano Di. and Harman, Mark. Search-Based Techniques Applied to Optimization of Project Planning for a Massive Maintenance Project. ICSM, page 240-249. IEEE Computer Society, (2005)
[26] L. Davis. Job-shop scheduling with genetic algorithms. In International Conference on GAs, pages 136–140. Lawrence Erlbaum, 1985
[27] E. Falkenauer. Genetic Algorithms and Grouping Problems. Wiley-Inter Science, Wiley - NY, 1998.
[28] A. Bagnall, V. Rayward-Smith, and I. Whittley. The next release problem. Information and Software Technology, 43(14):883–890, Dec. 2001.
[29] J. Aguilar-Ruiz, I. Ramos, J. C. Riquelme, and M. Toro.An evolutionary approach to estimating software development projects. Information and Software Technology, 43(14):875–882, Dec. 2001.
[30] G. Antoniol, M. Di Penta, and M. Harman. A robust \ search–based approach to project management in the presence of abandonment, rework, error and uncertainty. In 10th InternationalSoftware Metrics Symposium (METRICS 2004), pages 172–183, Los Alamitos, California, USA, Sept. 2004.IEEE Computer Society Press.
[31] G. Antoniol, M. D. Penta, and M. Harman. Search-based techniques applied to optimization of project planning for amassive maintenance project. In 21st IEEE International Conference on Software Maintenance, pages 240–249, LosAlamitos, California, USA, 2005. IEEE Computer Society Press.
[32] C. J. Burgess and M. Lefley. Can genetic programming improve software effort estimation? A comparative evaluation.Information and Software Technology, 43(14):863– 873, Dec. 2001.
[33] J. J. Dolado. A validation of the component-based method for software size estimation. IEEE Transactions on Software Engineering, 26(10):1006–1021, 2000.
[34] C. Kirsopp, M. Shepperd, and J. Hart. Search heuristics, case-based reasoning and software project effort prediction.In GECCO 2002: Proceedings of the Genetic and Evolutionary Computation Conference, pages 1367–1374, SanFrancisco, CA 94104, USA, 9-13 July 2002. Morgan Kaufmann Publishers.
[35] A. Baresel, D. W. Binkley, M. Harman, and B. Korel. Evolutionary testing in the presence of loop–assigned flags: A testability transformation approach. In International Symposium on Software Testing and Analysis (ISSTA 2004), pages 108–118, Omni Parker House Hotel, Boston, Massachusetts, and July 2004. Appears in Software Engineering Notes, Volume 29, and Number 4.
[36] A. Baresel, H. Sthamer, and M. Schmidt. Fitness function design to improve evolutionary structural testing. In GECCO 2002: Proceedings of the Genetic and Evolutionary Computation Conference, pages 1329–1336, San Fran-12 cisco, CA 94104, USA, 9-13 July 2002. Morgan Kaufmann Publishers.
[37] L. Bottaci. Instrumenting programs with flag variables for test data search by genetic algorithms. In GECCO2002: Proceedings of the Genetic and Evolutionary Computation Conference, pages 1337–1342, New York, 9-13 July2002. Morgan Kaufmann Publishers. Seattle, Washington, USA, 8-12 July 2006. ACM Press.
[38] L. C. Briand, J. Feng, and Y. Labiche. Using genetic algorithms and coupling measures to devise optimal integration orders. In SEKE, pages 43–50, 2002.
[39] L. C. Briand, Y. Labiche, and M. Shousha. Stress testing real-time systems with genetic algorithms. In Genetic andEvolutionary Computation Conference, GECCO 2005, Proceedings, Washington DC, USA, June 25-29, 2005, pages 1021–1028. ACM, 2005.
[40] Q. Guo, R. M. Hierons, M. Harman, and K. Derderian. Constructing multiple unique input/output sequences using evolutionary Optimization techniques. IEE Proceedings — Software, 152(3):127–140, 2005.
[41] M. Harman, L. Hu, R. M. Hierons, J. Wegener, H. Sthamer, A. Baresel, and M. Roper. Testability transformation. IEEETransactions on Software Engineering, 30(1):3–16, Jan.2004.
[42] Z. Li, M. Harman, and R. Hierons. Meta-heuristic search algorithms for regression test case prioritization. IEEE Transactions on Software Engineering. To appear.
[43] P. McMinn, M. Harman, D. Binkley, and P. Tonella. The species per path approach to search-based test data generation.In International Symposium on Software Testing and Analysis (ISSTA 06), pages 13–24, Portland, Maine, USA. 2006.
[44] J. Wegener, A. Baresel, and H. Sthamer. Evolutionary test environment for automatic structural testing. Informationand Software Technology Special Issue on Software Engineering using Metaheuristic Innovative Algorithms,43(14):841–854, 2001.
[45] S. Bouktif, G. Antoniol, E. Merlo, and M. Neteler. A novel approach to optimize clone refactoring activity. In GECCO 2006: Proceedings of the 8th annual conference on Genetic and evolutionary computation, volume 2, pages 1885–1892,Seattle, Washington, USA, 8-12 July 2006. ACM Press.
[46] D. Fatiregun, M. Harman, and R. Hierons. Search-based amorphous slicing. In 12th International Working Conference on Reverse Engineering (WCRE 05), pages 3–12, Carnegie Mellon University, Pittsburgh, Pennsylvania, USA, and Nov. 2005.
[47] M. Harman, R. Hierons, and M. Proctor. A new representation and crossover operator for search-based optimization of software modularization. In GECCO 2002: Proceedings of the Genetic and Evolutionary Computation Conference, pages 1351–1358, San Francisco, CA 94104, USA, 9-13 July 2002. Morgan Kaufmann Publishers.
[48] B. S. Mitchell and S. Mancoridis. Using heuristic search techniques to extract design abstractions from source code. In GECCO 2002: Proceedings of the Genetic and Evolutionary Computation Conference, pages 1375–1382, San Francisco, CA 94104, USA, 9-13 July 2002. Morgan Kaufmann Publishers.
[49] B. S. Mitchell and S. Mancoridis. On the automatic modularization of software systems using the bunch tool.IEEE Transactions on Software Engineering, 32(3):193–208, 2006.
[50] M. O'Keeffe and M. O'Cinneide. Search-based software maintenance. In Conference on Software Maintenance

and Reengineering (CSMR'06), pages 249–260, Mar. 2006.

[51] O. Seng, M. Bauer, M. Biehl, and G. Pache. Search based improvement of subsystem decompositions. In H.-G. Beyer and U.-M. O'Reilly, editors, Genetic and Evolutionary Computation Conference, GECCO 2005, Proceedings,Washington DC, USA, June 25-29, 2005, pages 1045–1051.ACM, 2005.

[52]O. Seng, J. Stammel, and D. Burkhart. Search-based determination of refactoring for improving the class structure of Object-oriented systems. In GECCO 2006: Proceedings of the 8th annual conference on Genetic and evolutionary computation, volume 2, pages 1909–1916, Seattle, Washington,USA, 8-12 July 2006. ACM Press.

[53] G. Canfora, M. D. Penta, R. Esposito, and M. L. Villani. An approach for QoS-aware service composition based on genetic algorithms. In H.-G. Beyer and U.-M. O'Reilly, editors, Genetic and Evolutionary Computation Conference,GECCO 2005, Proceedings, Washington DC, USA, June 25-29, 2005, pages 1069–1075. ACM, 2005.

[54] M. Cohen, S. B. Kooi, and W. Srisa-an. Clustering the heap in multi-threaded applications for improved garbagecollection. In GECCO 2006: Proceedings of the 8th annual conference on Genetic and evolutionary computation, volume2, pages 1901–1908, Seattle, Washington, USA, 8-12 July 2006. ACM Press.

[55] K. D. Cooper, P. J. Schielke, and D. Subramanian. Optimizing for reduced code space using genetic algorithms. In Proceedings of the ACMSigplan 1999 Workshop on Languages, Compilers and Tools for Embedded Systems (LCTES'99),volume 34.7 of ACM Sigplan Notices, pages 1–9, NY, May 5 1999. ACM Press.

[56] S. Bouktif, H. Sahraoui, and G. Antoniol. Simulated annealing for improving software quality prediction. In GECCO 2006: Proceedings of the 8th annual conference on Genetic and evolutionary computation, volume 2, pages 1893–1900,Seattle, Washington, USA, 8-12 July 2006. ACM Press.

[57] T.M. Khoshgoftaar, L. Yi, and N. Seliya. A multiobjective module-order model for software quality enhancement.IEEE Transactions on Evolutionary Computation, 8(6):593– 608, December 2004.

[58] H. H. Ammar, W. Abdelmoez, and M. S. Hamdi,"Software Engineering Using Artificial Intelligence Techniques: Current State and Open Problems", Proceedings of the First Taibah University International Conference on Computing and Information Technology (ICCIT 2012), Al-Madinah Al-Munawwarah, Saudi Arabia, 12-14 March 2012.

[59] Jonathan onowakpo goddey ebbah, deploying Artificial intelligence technique in Software engineering, American Journal of Undergraduate Research, VOL. 1 NO. 1, Department of Computer Science, University of Ibadan, Nigeria, Page: 19-24, 2002.

[60] Jain Prince, Interaction between Software Engineering and Artificial Intelligence- A Review , International Journal on Computer Science & Engineering; December 2011, Vol. 3 Issue 12, p3774.

[61] Raza, Farah Naaz, Artificial Intelligence Techniques in Software Engineering (AITSE), International Multi Conference of Engineers & Computer Scientists; January 2009, p1086 .

[62] Roger S. Pressman, Software Engineering: A Beginner's Guide (McGraw Hill Higher education Publishers, New York,New York, USA) 1988. [63]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,USA) 1988.

[64] Shari Lawrence Pfleeger, Software Engineering: theory and Practice (Prentice Hall Publishers, Upper Saddle River, New Jersey, USA) 1998. [65]Harman Mark, The Role of Artificial Intelligence in Software Engineering, CREST Centre, University College London, Malet Place, London, WC1E 6BT, UK.

[66] Yadav Nikita, Sujata Khatri ,Singh VB, Developing an Intelligent cloud for Higher Education, ACM SIGSOFT Software Engineering Notes, Volume 39 Issue 1, January 2014,Pages 1-5, ACM New York, NY, USA

[67] Yadav Nikita, Singh V B, Kumari Madhu, Generalized Reliability Model for Cloud Computing, International Journal of Computer Applications (0975 – 8887) Volume 88 – No.14, February 2014

[68] Thanadesh Thanakornworakij, Raja F. Nassar, Chokchai Leangsuksun, and Mihaela Păun: A Reliability Model for Cloud Computing for High Performance Computing Applications. In Springer-Verlag Berlin Heidelberg 2013, Euro-Par 2012 Workshops, LNCS 7640, pp. 474–483, 2013.

[69] Sujata Khatri, R.S.Chhillar, V.B.Singh: Measuring Bug Complexity in Object Oriented Software System. In ACM SIGSOFT Software Engineering Notes, volume 36 Issue 6, November 2011, pages 1-8.

[70] Dai, Y.S., Yang, B., Dongarra, J., Zhang, G.: Cloud Service Reliability: Modeling and Analysis. In: The 15th IEEE Pacific Rim International Symposium on Dependable Com- puting (2009).