International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 6s
DOI: https://doi.org/10.17762/ijritcc.v11i6s.6944
***Article Received**: 28 March 2023 **Revised**: 12 May 2023 **Accepted**: 31 May 2023*

Framework for the Automation of SDLC Phases using Artificial Intelligence and Machine Learning Techniques

Sahana P. Shankar[1], Shilpa Shashikant Chaudhari[2]
[1]Department of Computer Science and Engineering
Ramaiah University of Applied Sciences
Bengaluru, India
e-mail: sahanaprabhushankar@gmail.com
[2]Department of Computer Science and Engineering
M S Ramaiah Institute of Technology (Affiliated to VTU)
Bengaluru, India
e-mail: shilpasc29@msrit.edu

Abstract— Software Engineering acts as a foundation stone for any software that is being built. It provides a common road-map for construction of software from any domain. Not following a well-defined Software Development Model have led to the failure of many software projects in the past. Agile is the Software Development Life Cycle (SDLC) Model that is widely used in practice in the IT industries to develop software on various technologies such as Big Data, Machine Learning, Artificial Intelligence, Deep learning. The focus on Software Engineering side in the recent years has been on trying to automate the various phases of SDLC namely- Requirements Analysis, Design, Coding, Testing and Operations and Maintenance. Incorporating latest trending technologies such as Machine Learning and Artificial Intelligence into various phases of SDLC, could facilitate for better execution of each of these phases. This in turn helps to cut-down costs, save time, improve the efficiency and reduce the manual effort required for each of these phases. The aim of this paper is to present a framework for the application of various Artificial Intelligence and Machine Learning techniques in the different phases of SDLC.

Keywords-Requirements Elicitation, Knowledge Bank, Testing, Software Maintenance, Chatbot, Design.

I. INTRODUCTION

Software Engineering (SE) can be defined as application of engineering skills to the development of a software product. Over the years engineers have spent much time in building more intelligent software. With the increase in the level of intelligence associated with the software, the level of complexity also seems to be increasing. Complex software again poses new challenges to the software engineers at each of the phases of the software development life cycle (SDLC). Where developing complex system is a challenging task to the software engineers, developing intelligent ways of building the complex intelligent systems can be considered to a bigger challenge in itself. If a software engineer is able to achieve the latter task, it could as well simplify and aid in improving the efficiency of the earlier task. The Figure 1 below shows the relationship between Planning, Decision and Searching in terms of AI, ML and SE where SE involves more of planning, AI involves searching and ML involves decision making.

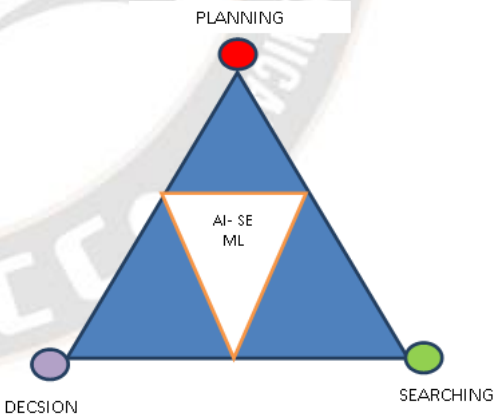

Fig 1. Connectivity between AI-ML-SE

Artificial Intelligence has gained a lot of popularity in the recent years in the various fields of Automotive, Banking, Medicine, Retails and Service Industry. It has been observed that AI has made its way into mundane everyday tasks of people in day to day life. With rapid growth and extensive research work being carried out in the field of AI and ML at an exponential rate, these results can be used to improve the Software Engineering Process.

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 6s
DOI: https://doi.org/10.17762/ijritcc.v11i6s.6944
***Article Received**: 28 March 2023 **Revised**: 12 May 2023 **Accepted**: 31 May 2023*

The different phases of SDLC on a high level can be broadly classified as Requirement Analysis, Design, Implementation, Testing, Operation and Maintenance. These phases will be present in any Software development model in additional to a few more phases. The Requirements phase involves mainly elicitation of the business requirements from the clients or stake holders through personal interviews or brainstorming. These requirements that are collected in Natural Language such as English are then converted to a more formal representation of data for the Software Requirements Specification Document. The conversion from Informal English sentences to more Formal Document is generally carried out by a Business Analyst who is a domain expert of the project. Alongside, with the development of SRS document, another document called as Risk Analysis and Mitigation document is prepared by making use of the Project and Process Metrics. A decision is also made as to what Requirements can be actually implemented and which cannot be.

This SRS document then acts as an input for the Design Phase, where the Formal Requirements are then converted to High Level Architectural Design and more detailed Low-Level Design Patterns. This job is carried out by the Software Architect. The selection of Appropriate Architectural Design or Design pattern is based on various Software Metrics and Measurement Parameters such as Halsted and McCabe's Metrics, Chidamber and Kerner (CK) Metrics to name a few. The output of this phase is the detailed Design Document.

The Detailed Document is then converted to a Computer Program or Code using various Product Metrics. This job is carried out by a Software Developer, who decides on which programming language and programming styles to be adopted. The output of this phase is working source code and the Software Application or Product. The developer also decides on the White Box Testing Techniques to be adopted.

The Source Code and the Software Application is then passed onto the Testing Phase, where a dedicated Testing team performs the Black Box Testing. The Quality Engineer who carries out this activity designs the test cases based on various Black Box Testing Techniques that are available. The Test Plan also needs to be designed at the beginning of the testing activity based on project Metrics by the Test Team Lead. This plan contains important decision as to which test cases need to be tested as a part of Regression Testing. The test cases then need to be executed either manually or using Automation Tools. Any bugs identified during this phase is generally logged in a Bug Report and then reverted back to the Development Team.

The final Quality Software is then released to the customers for use. The phase that comes into action now is the Operations and Maintenance Phase during which support will be extended to the end users or clients if any issues arise during the use of the software. The companies generally have a dedicated Support Team that cater to these issues by logging a ticket for every concern raised by the client.

The Figure 2 shows the different phases of SDLC which are found in any of the Software Development Models in terms of process model to be followed.

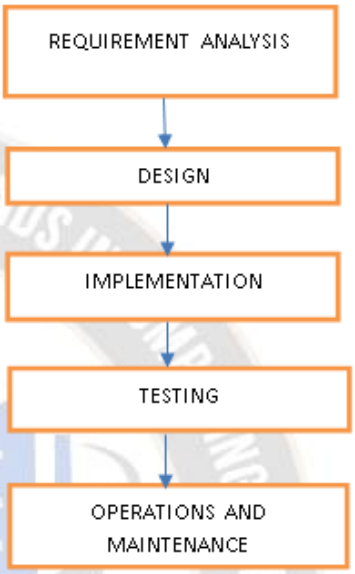

Fig 2. Overview of Different Phases of SDLC

II. RELATED WORK

This section discusses on the various works that has already been carried out in application of machine learning and AI techniques for improvising the different phases. In [1] the authors talk about applying supervised machine learning techniques to automate and address some the issues during the requirements engineering phase. It also discusses the datasets that have been used to carry out these experiments. The PROMISE data repository, Metric Data Program and iTrust electronic healthcare systems are used for the datasets. The problems addressed include linguistic problem detection in the SRS document that is written in natural language, application of classification algorithms to classify the contents of SRS document, defect traceability to the requirements, software effort estimation, prediction of failures using SRS and generation of business rules. The machine learning techniques used are random forest, Naïve Bayes, Support Vector Machine, decision tree and K-nearest neighbor. In [2] author discusses that uncertainty is an integral part of any software, especially the embedded softwares involving robots, automated cars or unmanned vehicles. Customer satisfaction being the important goal of any software being developed, it is also important to address these uncertainties and ensure software compliance. The important requirements include security and privacy among others. This paper also addresses the security and privacy

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 6s
DOI: https://doi.org/10.17762/ijritcc.v11i6s.6944
***Article Received**: 28 March 2023 **Revised**: 12 May 2023 **Accepted**: 31 May 2023*

concerns of the software by proposing a framework for handling uncertainty. In [3] the authors proposes a tool that can automatically convert the elicited requirements in a technical format. It also provides an option to collect the end user feedback. The manual way of doing this slows down the process and has an impact on usability and reliability. The tool uses supervised machine learning and unsupervised deep learning strategies. The supervised machine learning algorithms used include Logistic, LogitBoost, Linear Regression, J48, and AdaBoostM1. In [4] the author talks about generation of use cases from the elicited requirements. The requirements collected will be in natural language and hence it is difficult to comprehend the dependencies, inconsistencies or any missing requirements. The very first UML diagram that is generally drawn is the Use Case that depicts the interaction of the user with the outside world. It helps to identify the different stakeholders interacting with the software system. Such a graphical document is very easy to comprehend and understand. The diagram also depicts the dependencies between the use cases using relationships such as include and extend. This takes considerable amount if effort if it needs to be done manually. This process when automated with the proposed machine learning techniques has proven shorten the time taken for generation. In [5] the authors discuss about the difficulties in eliciting the requirements for a machine learning based software. They suggest a new strategy called as Goal Oriented Software Requirements Engineering to collect the requirements. In [6] the authors discuss about implementing one of the most important characteristic of good requirements i.e. traceability. He talks about using trace matrix or providing the connection between the various requirements. This work also has been carried out using the datasets from the PROMISE repository. In [7], the authors talk about the challenges involved in carrying out the requirements engineering phase for machine learning based projects. Specific legal requirements, explain ability and freedom from discrimination are the new techniques adopted in the requirements engineering process. In [8] the authors talk about improving the efficiency of software process models using AI techniques. They implement IoT based methods to automate the existing process with minimum disruption. They also provide the datasets for researchers to continue the research further. They have suggested the use of various unsupervised machine learning algorithms for the same. In [9] the author yet again discusses about ensuring the good characteristics of the requirements, more specifically assurance of requirements quality and validating the same. Detection and correction of the errors in the requirements is an important aspect of this paper. Some of the techniques include formal methods, prototyping, test oriented, and knowledge oriented validation among others. The latest trend is to perform validation using Machine Learning techniques. In [10] the authors provide the recommendations and suggestions to the practitioners on the various machine learning algorithms that can be used across the various phases, based on the survey conducted by them. In [11] the authors propose a new tool called "BUDGET" that helps in the high level design phase of software engineering. They propose the use of supervised machine learning algorithms for the same. By using these techniques the user can bridge the gap between requirements, architecture and code. In [12] the authors propose a testing framework for validating the software products based on machine learning techniques. Testing of ML based software products is much difficult than the non-ML counterparts mainly because of self-learning and rapid evolution capability of ML algorithms. Hence, there must be special measures that would be considered for these products. Robustness, avoid ability, achievability, improvability and allow ability are the features taken into consideration for evaluating the performance of the ML based software products. In [13] the authors propose how to use machine learning techniques for Agile software development process. They talk about automating the client feedback received during the retrospective meeting to improve the Agile development process continuously. They use sentiment analysis for this purpose. In [14] the authors handle a very important aspect i.e. software project effort estimation which needs to be performed during the project inception. This paper proposed "Extreme Learning Machine" model to perform the effort estimation. It is suggested that Particle Swarm Optimization technique can be used to optimize this further. In [15] the authors conduct a literature survey on what algorithms are used in what phases of the software development life cycle. It was observed that most of applications of the machine learning algorithm is mainly for the testing and quality assurance phase, design and architecture, development, requirements engineering and maintenance in that order. It was also observed that Artificial Neural Network (ANN) is the most widely applied algorithm and Linear Regression being the least preferred technique. In [16], the author talks about applying machine learning technique to the implementation phase. He suggests how to detect vulnerable or faulty code in the software. He uses classification techniques to separate vulnerable code from non-vulnerable ones. Decision Tree, Support Vector Machine, and Random Forest to do the job. They used the projects developed using C++/C. Software metrics are used along with machine learning techniques to do the job. In [17] the authors talk about software size estimation using the machine learning model deep learning. This is achieved using software metrics called as Function Points (FP). He says that the manual way of estimating the software takes considerable amount of effort and also can be erroneous. This also leads to increase in the cost of software estimation. The deep learning based model called as "Named Entity Recognition" model does the job automatically. The experiments were conducted on the 29 live projects. The Table

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 6s
DOI: https://doi.org/10.17762/ijritcc.v11i6s.6944
***Article Received**: 28 March 2023 **Revised**: 12 May 2023 **Accepted**: 31 May 2023*

1 below summarizes the various machine learning techniques used in the different phases of Software Development Life Cycle [15]. In the software quality assurance phase, bug prediction and detection are the two most important aspects. After the bug has been detected or predicted, it is also important to categorize them into various categories based on priority and severity. The authors in [18] propose a method for doing this automatically using machine learning and natural language processing technique. This helps to identify and address the high priority bugs early. The Mozilla and Eclipse repository has been used for the experimental purpose. The classification algorithms Naïve Bayes, logistic regression, decision tree and random forest has been used for the experimental purpose. In the software development models like the Boehm's spiral model, risk mitigation is a very important factor. Identification and mitigation of risks is an important phase that happens in every iteration. In [19] the authors discuss a way of doing this automatically using machine learning techniques. They use association rule mining for identifying the associations between risks and mitigation strategies. This uses a hybrid approach consisting of Case based reasoning and Association Rule mining. Based on the data collected, they list down 26 different software risk factors and 50 different mitigation strategies to address the same [19]. In [20] the authors talk about the different ways of predicting the bugs in the software even before the quality assurance phase can begin. They make use of deep learning algorithms to achieve this task. These include the two algorithms namely Convolutional Neural Networks and Multi-Layer Perceptron. They have conducted the experiments on the NASA defect datasets namely CM1, KC1, KC2 and PC1. In [21] authors talk about the importance of documentation throughout all the phases of the software development life cycle. Bug duplication is also an important aspect that needs to be addressed during the testing phase. After the duplicate bugs are detected, the next step is using the classification algorithm to classify whether the bug is duplicate or not. These experiments were carried out on datasets provided by Eclipse, Mozilla, Firefox, NetBeans and Open office. In [22] the authors discuss about the growing complexity of the software and its influence on the quality of the product. There is no way that all the defects can be predicted beforehand using machine learning techniques. So we can never say that we have detected all the defects before the software enters the testing phase. The authors here propose a technique for predicting the failures using ensemble machine learning techniques. Research shows that ensemble techniques provide better result than the individual models. In [23], the authors talk about the bias that influence the experimental results in the field of defect prediction. The results show that there is a relationship between the research group who perform the task and the dataset that has been used. This needs to be addressed immediately to remove the bias in the results thus obtained. In [24] the authors use the 10 NASA defect datasets to perform defect prediction. They propose a novel technique called as "Hellinger Net Model" that is a deep forward neural network to perform the task. The experimental results show a drastic improvement in the performance measures. In [25] the authors propose a framework for predicting the faulty modules. This uses the historical data for predicting the faulty modules. The experiments have been conducted on the NASA defect datasets. Three different learning algorithms used were OneR, Naïve Bayes and J48 on twelve different datasets. In [26] the authors talk about the commonalities between the two fields of Software Engineering and Artificial Intelligence in aspects of modelling real world objects. He talks about role of Software Agents in Distributed Artificial Intelligence Systems. He speaks about role of Learning Software Organizations and Knowledge Based Systems. Computational Intelligence plays an important role in analysis of software and project management, knowledge discovery from existing databases. They talk about the current role and future trends in the intersection of these two areas where a major part of the research is being carried out by international conference and journal called "Automated Software Engineering". In [27] the author emphasizes on the importance of Search Based Software Engineering and application of Machine Learning for software engineering. Since there is a shift in focus from small, localized, insulated construction of software to more interactive, intelligent, real-time, complex systems, the application of AI to SE is a good option. He also proposes to intelligently balance automation and human intervention. He also talks about adopting search technique for test data to cover a specific requirements set. In [28] the authors say that by using Software Analytics, we can eliminate some assumptions made by developers on good and bad software on the basis of a few projects they have worked upon. He also speaks about the problem of context in Software Analytics where models learned from one project may not be applicable to a new project at hand. This is one open area of research. Yet another issue is that the models generated by software analytics are not always easy and direct to understand and read. The machine learning algorithms make use of the complex mathematical models such as Naïve Bayes Classifier, Random Forests which are hard to understand. The focus could be on developing the results from these models in a simplified manner that is easier to understand. In [29] the author focuses on shifting the use of human intelligence to model intelligent systems that can do the designing and other software engineering tasks rather than doing the software engineering task itself. The author talks on the synergy between co-operative testing, human and artificial intelligence. He talks about some of the example problems where AI can be applied such as Test Generation, Specification generation, Debugging and Programming. He also suggests on improving the AI algorithms, by taking using continuous feedback loop. In [30] the author

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 6s
DOI: https://doi.org/10.17762/ijritcc.v11i6s.6944
***Article Received**: 28 March 2023 **Revised**: 12 May 2023 **Accepted**: 31 May 2023*

focuses entirely on the Requirements Phase, where he says that Software Requirements Classification task requires a lot of effort and hence incorporating NLP and Information Retrieval techniques at this stage. This paper employs Deep Learning and Neural Networks to do the task. They also propose employing AI to do project cost and effort estimation by even considering team member capabilities. In [31] the authors talk about the application of ML to SE phases such as Behavior Extraction, Defect Fixing, and Testing. They also propose that with the rapid development in the field of AI, the performance of the algorithm improves every year because of the rapid amount of data accumulated over time which in turn improves the learning process. They also talk about the challenges of how scalability of the machine learning algorithms need to be handled with increase in software complexity. In [32] authors talk about employing Software Metrics with Defect Data to develop Predictive Models for Open Source Software Systems. The comparison of 14 different ML algorithms such as variants of Perceptron, Multi-Layer Perceptron, LVQ, SOM, AIRS, CLONAL and Immune for defect prediction is used. The study of the results show that Single layer Perceptron is the best defect prediction algorithm. In [33] authors talk about the ongoing work in ML and AI platforms and how these results can be applied to SE. The authors talk about the many open source tools that are available for ML that create motivation for students to work in this field. They talk about the essentials to develop algorithms in this field such as programming skills in R, Python with MatPlotLib. They also mention Jupyter Notebook which is an open-source web application that allows to creation and sharing of live-code, visualization and text. In [34] authors use three machine learning algorithms namely Logistic Regression, Naïve Bayes and J48 on several datasets to perform defect prediction. The experimental results proved that using reduced metrics set is better than using the entire metric set to generate the results. G-score and recall provide good results despite using full set of metrics. Among the algorithm tested on reduced set of metrics, J48 outperformed others using G-score. Datasets that were used were from open source eclipse releases. The Table 1 lists the AI/ML algorithm used in SDLC phases based on studied literature.

Table 1. Summary of algorithms used based on literature review

PHASE IN SDLC	ML ALGORITHMS USED
Requirements Engineering	Support Vector Machines, Naïve Bayes, Random Forest, Decision Tree, K-Nearest Neighbor, Linear Regression, AdaBoost, Logistic Regression, Natural Language Processing, Recurrent Neural Network, and Long Short-Term Memory
Software Design	Natural Language Processing, Decision Tree, Multinomial Naïve Bayes, Perceptron, Linear Classifier, Passive Aggressive Classifier, Vector Space Machines, Recurrent Neural Network, Random Forest, Linear Regression, Principal Component Analysis, Artificial Neural Network, Genetic Algorithm
Software Development	Convolution Neural Networks, Recurrent Neural Network, Natural Language Processing, Artificial Neural Network, Decision Tree, Naïve Bayes, Support Vector Machine, Linear Regression, AdaBoost, J48, Extreme Gradient Boosting
Software Testing	Recurrent Neural Network, Decision Tree, Natural Language Processing, Naïve Bayes, Artificial Neural Network, Convolutional Neural Network, Support Vector Machine, Tree Augmented Naïve Bayes, KStar, Genetic Algorithm, Clustering, K-Means, Long Short Term Memory, Association Rule Mining, Multi-Layer Perceptron
Software Maintenance	Support Vector Machine, Recurrent Neural Network, Decision Tree, Natural Language Processing, Naïve Bayes, K Nearest Neighbor, Sequential Minimal Optimization, Federated Learning

III. PROPOSED METHODOLOGY

The proposed framework in the Figure 3 shows how AI and ML techniques can be adopted to replace some of the Mundane Tasks that are carried out in each phase of SDLC. The ML and AI techniques used in the proposed novel framework are summarized in the Table 2. It also proposes some innovative ways of adopting AI and ML to make the flow of the tasks in each of these phases smoother. If the company has been qualified as a CMMI level company or adheres to the ISO standards, then the standard process or template can be stored in the KB at each of the levels of SDLC, so that the algorithm is able to make sure quality is met each time.

A Requirements Analysis Phase

The Requirements Gathering and Engineering is a very important phase in the entire SDLC since it is the inception stage, from which the project kicks off. The success at this stage greatly impacts the success of overall project. This contains the information in an unstructured format that acts as main source of input to the Software Architects in the Design Phase. Some of the common problems in the Requirements Engineering Phase includes, Poor Requirements Quality where the requirements are ambiguous, incomplete, inconsistent, and incorrect and obsolete [1]. One of the major causes of any of these problems is lack of expertise or inadequate training of the Requirement Engineers. This dependency on the personnel can be eliminated by making use of a CHATBOT-an AI chat agent as shown in the fig 2. This

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 6s
DOI: https://doi.org/10.17762/ijritcc.v11i6s.6944
***Article Received**: 28 March 2023 **Revised**: 12 May 2023 **Accepted**: 31 May 2023*

CHATBOT makes use of the Knowledge Base which is mainly constructed from the Domain Knowledge of the Project under consideration like Banking, Retail, HealthCare, Finance, Travel, and Entertainment. This KB in turn consists of two parts, Knowledge Representation and Inference Engine. The CHATBOT also needs to make use of a NLP Analyzer to parse the sentences posed by the end-users and understand them in order to take necessary actions. Here the KB consists of the domain specific ontology definitions, common vocabulary as to how to represent the data, all the previous chat history of conversation with its end-users or stake-holders, and also some frequently asked questions related to that domain. This KB must be rich with data in order to facilitate the CHATBOT to make intelligent decisions. The CHATBOT can also be implemented to take input in the form of audio from the clients. Since most of requirements are unstructured and gathered in free-flow English, the conversion from unstructured form to a more formal representation depends entirely on the knowledge base and experience of the Requirements Engineer. Having a CHATBOT in place of a Requirement Engineer can eliminate the dependency on human personnel to get the task done. It is also seen that different dimensionally. If you must use mixed units, clearly state the units for each quantity that you use in an equation people interpret the same requirement in different ways. This leads to conflict of interest if more Requirement Engineers are participating in the task of eliciting requirements. Also on many occasions, the quality of the requirement elicitation depends on the emotional mood, health and temper of the requirement engineer, on the contrary an AI CHATBOT, it is very stable. Once the requirements are elicited by the CHATBOT, they are converted to a more formal representation and added to the SRS document. The SRS document is also updated further based on the second KB pertaining to the domain knowledge of the project, details on project team, and archives of the previous projects carried out in the same domain. A Classification Algorithm, can be implemented to identify and classify the requirements as Function or Non-Functional Requirements. The Risk Analysis and Mitigation is another important part of SDLC. This is incorporated at various stages in different ways depending on the Software Development Model used. The main aim of this stage, is to identify the potential risks that might come-up in the project based on the new requirements gathered along with the existing requirements and the other project details present in the KB. These details are then updated to the SRS document. Various machine learning Prediction Algorithms can be incorporated at this stage. The next important phase is to do the Metric Calculation and Cost Estimation of the project. This is very important very any software project and there are some well-defined software metrics standards to carry out this task. Here again some Machine Learning Algorithms can be designed to perform the Estimation and Prediction activities. This takes the input from the existing data in the KB along with the SRS document. Also a Recommender System can be implemented based on the SRS Document and the existing KB to suggest which of the new requirements can be practically implemented given the various factors and circumstances and which cannot be. This also identifies the non-testable requirements. Provides suggestions on any incomplete logic that is present. From the existing KB, the ML algorithm will be able to learn those requirements which when implemented in the past have led to the failure or the project or have not been able to reach the end-user.

B Design Phase

The SRS Document acts as input to the design phase, based on the new requirements incorporated, a ML algorithm can be used to recommend the suitable High-Level Architectural Design and Low-Level Design patterns. The ML Algorithms recommends the Functions, Modules and Interfaces that can be implemented based on the Requirements in the SRS Document. It also generates a skeletal Entity-Relationship Diagram, Flow-Charts indicating the flow of entire process.

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 6s
DOI: https://doi.org/10.17762/ijritcc.v11i6s.6944
Article Received: 28 March 2023 Revised: 12 May 2023 Accepted: 31 May 2023

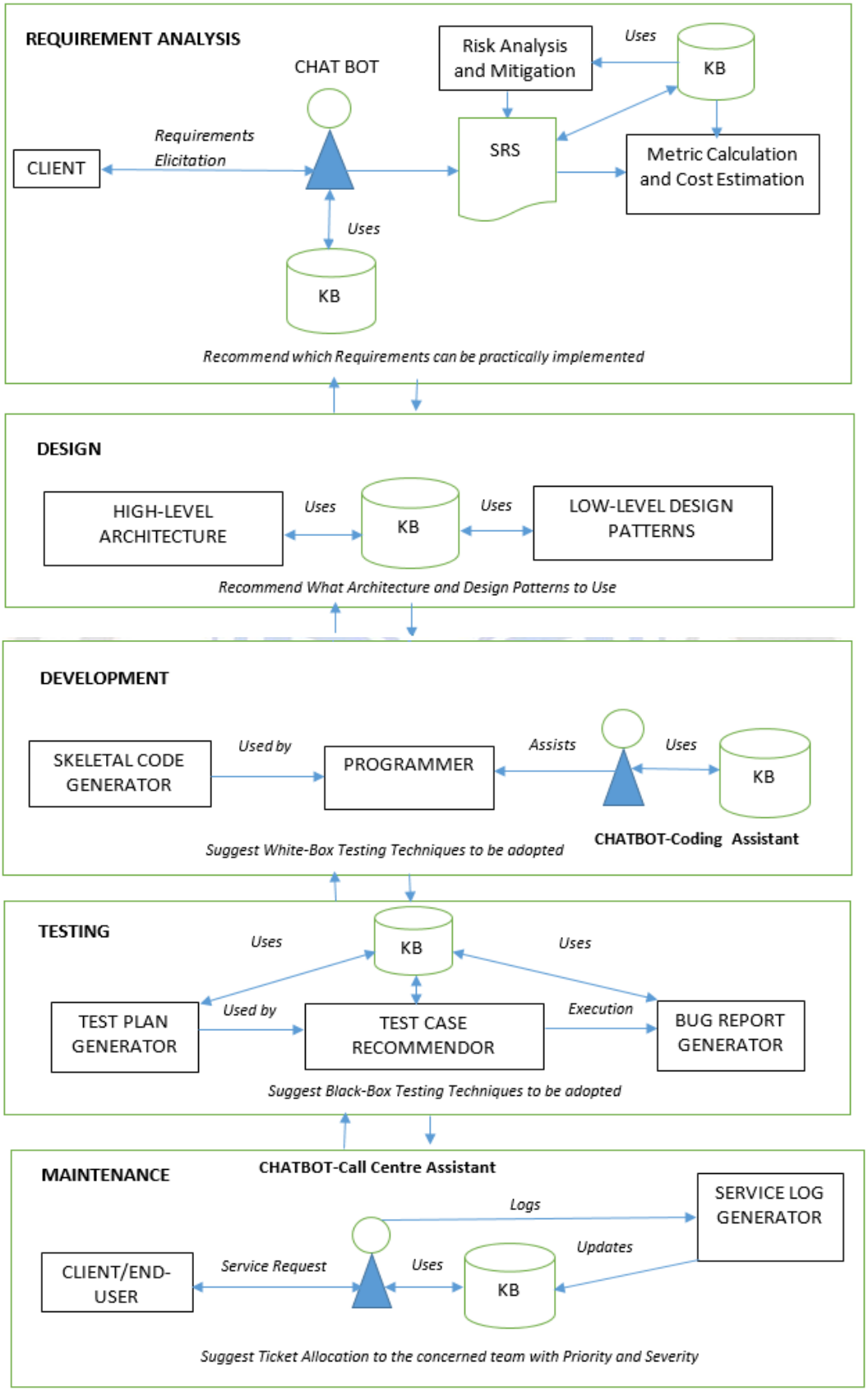

Fig 3. Proposed Novel Framework

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 6s
DOI: https://doi.org/10.17762/ijritcc.v11i6s.6944
Article Received: 28 March 2023 Revised: 12 May 2023 Accepted: 31 May 2023

Table 2.Summary of activities for the proposed framework

PHASE	ACTIVITY	ML TECHNIQUES	AI TECHNIQUES
Requirements Analysis	Requirements Elicitation-Interview Requirements Classification Risk Mitigation and Analysis	Classification Algorithms, Prediction Algorithms	CHATBOT-Requirements Elicitation
Software Design	High-Level Architectural Design Low-Level Detailed Design	Classification Algorithms, Prediction Algorithms	
Software Development	Skeletal Code Generator Trouble Shoot while Coding Suggest White-Box Testing techniques.	Classification Algorithms, Prediction Algorithms	CHATBOT-Coding Assistant,
Software Testing	Test Plan Generator Test Case Recommender Bug Report Generator	Classification Algorithms, Clustering, Prediction Algorithms	
Software Maintenance	Customer Support Service Log Generator	Classification Algorithms	CHATBOT-Support Representative

The Low-Level Detailed Design involves employing a ML algorithm to suggest the design patterns based on the interaction between the modules, functions and interfaces identified in the High-Level Design. The Design Engineers and Software Architects can make use of these inputs and further make modifications to it in more detail and create a solid Design Document. This phase makes use of a separate KB containing the information needed to support the design.

C Implementation Phase

The Design Document acts as an input to the Coding Phase, where a software can be developed to convert the design to a skeletal code. The developer then can make use of this skeletal code and the work on it further to write a more detailed code. A CHATBOT-Coding Assistant is employed in this phase that provides assistance to the developer for developing the code. During the code phase, if the developer gets stuck in any part while developing the code, by contacting the CHATBOT it provides tips as to how to tackle with the problem by providing suggestions which is again derived from a KB which is a rich repository of common problems encountered by the developers in the past. In addition to this, it can also provide links to the websites from around the web that are most commonly consulted by the developers while developing the code. Prior to this the KB, can also be loaded with information on the team members and developers who have worked on that piece of code in past, and the AI-CHATBOT can provide suggestions on whom to contact. In addition to this a Machine Learning Prediction Algorithm can be developed to predict the coding mistakes before the developer makes and prevent them.

C Quality Assurance Phase

The Design Document acts as an input to the Coding Phase, where a software can be developed to convert the design to a skeletal code. The developer then can make use of this skeletal code and the work on it further to write a more detailed code. A CHATBOT-Coding Assistant is employed in this phase that provides assistance to the developer for developing the code.

During the code phase, if the developer gets stuck in any part while developing the code, by contacting the CHATBOT it provides tips as to how to tackle with the problem by providing suggestions which is again derived from a KB which is a rich repository of common problems encountered by the developers in the past. In addition to this, it can also provide links to the websites from around the web that are most commonly consulted by the developers while developing the code. Prior to this the KB, can also be loaded with information on the team members and developers who have worked on that piece of code in the past, and the AI-CHATBOT can provide suggestions on whom to contact. In addition to this a Machine Learning Prediction Algorithm can be developed to predict the coding mistakes before the developer makes and prevent them. The fully developed and functional application acts as an input to the Quality Team, where Test Plan Generator is employed to generate the Test Plan for the current cycle of SDLC. It makes use of the KB present in this phase that contains the details on the personnel, schedule, history of previous defects, test case repository i.e. test case management tool, Requirements from SRS. The Test Plan is a standard template document that contains the set of test cases to be executed from existing Test bed based on the Requirements and the Code. The Test Case development is one of the activities that takes a significant percentage of verification and validation efforts. The Test Case

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 6s
DOI: https://doi.org/10.17762/ijritcc.v11i6s.6944
***Article Received**: 28 March 2023 **Revised**: 12 May 2023 **Accepted**: 31 May 2023*

Recommender takes the input from the KB of the Test Phase as well as Test Plan that is generated by the Test Plan Generator. It is practically impossible to run nearly lakhs of tests in the Test Bed. It is necessary to identify the most likely functions to get impacted and run optimal regression tests. This Recommender makes use of the Machine Learning Prediction algorithm along with historical defect data present in the KB to build a prediction model that is able to capture the new data and predict what test cases need to be executed from existing test bed as well as develop new test cases for the missing scenarios based on the new requirements. After the Recommended Test Cases are executed, the results of Test Case Execution are passed onto the Bug Report Generator. This Bug Report Generator makes use of KB to generate a Bug Report in a standard template format as decided by the Quality Team. This Bug Report also indicates the Severity, Priority of each Bug reported along with assigning the bug to the respective developer or concerned team.

D Operations and Maintenance Phase

After the tested quality software has been delivered to the customer/client/end-user for use, if they encounter any problem during the operation phase, they can consult the customer care team of the company. Here the CHATBOT-Call Centre Assistant acts a point of contact to the customer. The client places the service request with the CHATBOT that makes use of a dedicated KB to try to address the problem. This KB consists of some frequently asked question-answers by customers, required amount of domain knowledge, information on the teams, knowledge on commonly encountered problems and workaround. The CHATBOT makes use of this rich knowledge base to log a ticket for every query of the customer. The CHATBOT also assigns the ticket to the concerned team/team manager with priority and severity. The service log generator generates a ticket based on the input from the CHATBOT. A standard template is agreed upon for the Service Log. Once the details are updated in this log by the CHATBOT, this is again updated in the KB to help in training of learning algorithms of ML that will be used in this phase.

IV. RESULTS

The results section here shows the pictorial representation of the framework represented using Figure 3 that has been described above in the method section. Each of these phases can be executed independently or can be implemented end to end for any software. The Table 3 above provides a matrix that indicates the coverage of AI and ML techniques across the various phases. The results section also discusses the current usage percentage of AI and ML algorithms across the different phases. These are indicated in the Figure 3. It gives an overview as to which algorithms are used across all the phases of Software Development Life Cycle (SDLC) and which algorithms are used only for a single phase of SDLC. Figure 4 is a bar graph representing the percentage usage of AI/ML algorithms across the different phases of SDLC. The X-Axis represents the various algorithms used for automating the phases of SDLC. Twenty five different algorithms are considered based on Table 1. Considering that SDLC has five phases, 100% in the above graph implies that the corresponding algorithms are used in all the five phases of SDLC. From the Figure 4, one can imply that DT, NB, NLP, RNN and SVM are used in all the phases of SDLC.

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 6s
DOI: https://doi.org/10.17762/ijritcc.v11i6s.6944
Article Received: 28 March 2023 Revised: 12 May 2023 Accepted: 31 May 2023

Table 3. Mapping of AI/ML Algorithm Usage for SDLC phases

ML Algorithms	Requirements	Design	Development	Testing	Maintenance
ADA Boost	√	-	√	-	-
ANN	-	√	√	√	-
-ARM	-	-	-	√	-
CNN	-	-	√	√	-
Clustering	-	-	-	√	-
DT	√	√	√	√	√
EGB	-	-	√	-	-
FL	-	-	-	-	√
GA	-	√	-	√	-
J48	-	-	√	-	-
KNN	√	-	-	-	√
K Star	-	-	-	-	-
LSTM	√	-	-	√	-
Linear Regression	√	√	√	√	-
Logistic Regression	√	-	-	-	-
NB	√	√	√	√	√
NLP	√	√	√	√	√
Perceptron	-	√	-	√	-
PCA	-	√	-	-	-
PAC	-	√	-	-	-
RF	√	√	-	-	-
RNN	-	√	√	√	√
SVM	√	-	√	√	√
SMO	-	-	-	-	√
VSM	-	√	-	-	-

The 60% in the Figure 4 corresponding to Artificial Neural Networks (ANN) and Linear Regression implies that these algorithms are applicable to three phases of SDLC. The corresponding phases can be identified from the matrix in the Table 3. Convolutional Neural Network (CNN), Clustering, Genetic Algorithm (GA), K-Nearest Neighbor (KNN), Long Short Term Memory (LSTM), Perceptron and Random Forest (RF) are being used in two phases of SDLC. Hence their percentage usage is 40%. Similarly Association Rule Mining (ARM), Extreme Gradient Boosting (EGB), Federated Learning (FL), K-Star, Logistic Regression, Principal Component Analysis (PCA), Passive Aggressive Classifier (PAC), Sequential Minimal Optimization (SMO) and Vector Space Machine (VSM) are used only in one of the phases of SDLC. Hence the usage is 20% across the phases of SDLC. A part of the framework in the Table 3 below has already been implemented and published as an IEEE conference paper in [35]. The automation of requirements elicitation phase of requirements engineering was carried out using an AI chatbot that elicited the requirements from the customer both in text mode and voice mode. It used the AI techniques Natural Language Processing (NLP) and Natural Language Generation. Machine Learning techniques such as Long Short Term Memory (LSTM) and Recurrent Neural Networks (RNN) were used. The classification algorithms such as Support Vector Machine (SVM) and Naïve Bayes (NB) were used for classification of functional requirements and non-functional requirements. A comparative analysis of these two classification algorithms were carried out and the results are indicated in the Table 4 given below.

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 6s
DOI: https://doi.org/10.17762/ijritcc.v11i6s.6944
***Article Received**: 28 March 2023 **Revised**: 12 May 2023 **Accepted**: 31 May 2023*

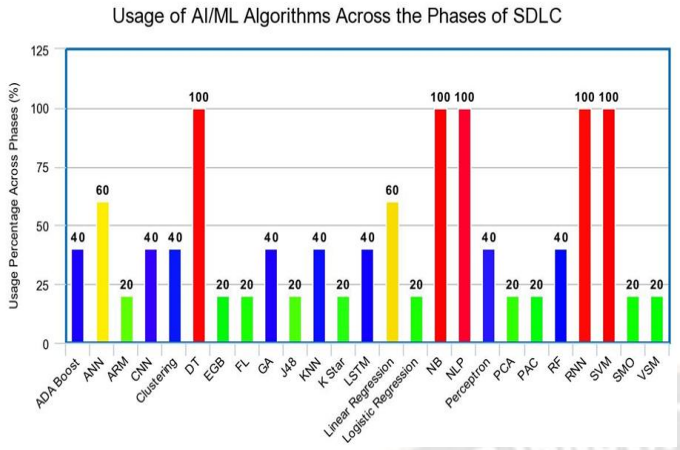

Fig 4. Percentage usage of AI/ML algorithms for the stages of SDLC

Table 4. Comparative Analysis of the Classification Techniques for Requirements Engineering Phase

Classification Technique	**Performance Metrics**			
	Accuracy	***Precision***	***Recall***	***F1-Score***
Naïve Bayes	0.91	0.91	0.91	0.91
SVM	0.88	0.88	0.88	0.88

V. CONCLUSION AND FUTURE WORK

Software engineering defines the process of developing a software that meets the customer requirements within the time and budget constraint. This is generally considered to be definition of successful software. But the definition of successful software is changing. A software can be a failure even if it meets all the requirements and is delivered on time and within budget because it simply fails to impress the customers. However, a software that is delivered with a few requirements and stretches a little beyond the deadlines can be considered as successful because it becomes a hit in the market release. What is more important here is to be able to deliver the same quality of work repeatedly to the end users each time irrespective of the project on hand. This can be achieved by having a framework in place such as the one defined in the paper which relies heavily of data archives and past experiences to cater better for the customer needs. The Machine Learning and Artificial Intelligence techniques are evolving rapidly and with the advent of new algorithms, small changes can be adopted to the framework simply by replacing the algorithms. As a part of the future work, more algorithms can be suggested to fully automate the process at various levels. Also customized frameworks can be proposed for individual process models.

REFERENCES

[1] Gramajo, M., Ballejos, L. and Ale, M., 2020. Seizing requirements engineering issues through supervised learning techniques. IEEE Latin America Transactions, 18(07), pp.1164-1184.

[2] Chechik, M., 2019, September. Uncertain requirements, assurance and machine learning. In 2019 IEEE 27th International Requirements Engineering Conference (RE) (pp. 2-3). IEEE.

[3] Panichella, S. and Ruiz, M., 2020, August. Requirements-collector: automating requirements specification from elicitation sessions and user feedback. In 2020 IEEE 28th International Requirements Engineering Conference (RE) (pp. 404-407). IEEE.

[4] Tiwari, S., Rathore, S.S., Sagar, S. and Mirani, Y., 2020, August. Identifying Use Case Elements from Textual Specification: A Preliminary Study. In 2020 IEEE 28th International Requirements Engineering Conference (RE) (pp. 410-411). IEEE.

[5] Ishikawa, F. and Matsuno, Y., 2020, August. Evidence-driven requirements engineering for uncertainty of machine learning-based systems. In 2020 IEEE 28th International Requirements Engineering Conference (RE) (pp. 346-351). IEEE.

[6] Dekhtyar, A. and Hayes, J.H., 2018. Automating requirements traceability: two decades of learning from KDD. arXiv preprint arXiv:1807.11454.

[7] Vogelsang, A. and Borg, M., 2019, September. Requirements engineering for machine learning: Perspectives from data scientists. In 2019 IEEE 27th International Requirements Engineering Conference Workshops (REW) (pp. 245-251). IEEE.

[8] Canedo, Arquimedes, Palash Goyal, Di Huang, Amit Pandey, and Gustavo Quiros. "ArduCode: Predictive Framework for Automation Engineering." IEEE Transactions on Automation Science and Engineering 18, no. 3 (2020): 1417-1428.

[9] Atoum, I., Baklizi, M., Alsmadi, I., Otoom, A.A., Alhersh, T., Ababneh, J., Almalki, J. and Alshahrani, S., 2021. Challenges of Software Requirements Quality Assurance and Validation: A Systematic Literature Review. IEEE Access.

[10] Wan, Z., Xia, X., Lo, D. and Murphy, G.C., 2019. How does machine learning change software development practices?. IEEE Transactions on Software Engineering, 47(9), pp.1857-1871.

[11] Santos, J.C., Mirakhorli, M., Mujhid, I. and Zogaan, W., 2016, April. BUDGET: A tool for supporting software architecture traceability research. In 2016 13th Working IEEE/IFIP Conference on Software Architecture (WICSA) (pp. 303-306). IEEE.

[12] Nishi, Y., Masuda, S., Ogawa, H. and Uetsuki, K., 2018, April. A test architecture for machine learning product. In 2018 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW) (pp. 273-278). IEEE.

[13] Tekbulut, T., Canbaz, N. and Kaya, T.Ö., 2022. Machine Learning Application in LAPIS Agile Software Development Process. In Intelligent Computing (pp. 1136-1148). Springer, Cham.

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 6s
DOI: https://doi.org/10.17762/ijritcc.v11i6s.6944
***Article Received**: 28 March 2023 **Revised**: 12 May 2023 **Accepted**: 31 May 2023*

[14] De Carvalho, H.D.P., Fagundes, R. and Santos, W., 2021. Extreme Learning Machine Applied to Software Development Effort Estimation. IEEE Access, 9, pp.92676-92687.

[15] Shafiq, S., Mashkoor, A., Mayr-Dorn, C. and Egyed, A., 2021. A Literature Review of Machine Learning and Software Development Life cycle Stages. IEEE Access.

[16] Medeiros, N., Ivaki, N., Costa, P. and Vieira, M., 2020. Vulnerable code detection using software metrics and machine learning. IEEE Access, 8, pp.219174-219198.

[17] Zhang, K., Wang, X., Ren, J. and Liu, C., 2020. Efficiency improvement of function point-based software size estimation with deep learning model. IEEE Access, 9, pp.107124-107136.

[18] Ahmed, H.A., Bawany, N.Z. and Shamsi, J.A., 2021. Capbug-a framework for automatic bug categorization and prioritization using nlp and machine learning algorithms. IEEE Access, 9, pp.50496-50512.

[19] Asif, M. and Ahmed, J., 2020. A novel case base reasoning and frequent pattern based decision support system for mitigating software risk factors. IEEE Access, 8, pp.102278-102291.

[20] Al Qasem, O., Akour, M. and Alenezi, M., 2020. The influence of deep learning algorithms factors in software fault prediction. IEEE Access, 8, pp.63945-63960.

[21] Kukkar, A., Mohana, R., Kumar, Y., Nayyar, A., Bilal, M. and Kwak, K.S., 2020. Duplicate bug report detection and classification system based on deep learning technique. IEEE Access, 8, pp.200749-200763.

[22] Campos, J.R., Costa, E. and Vieira, M., 2019. Improving failure prediction by ensembling the decisions of machine learning models: A case study. IEEE Access, 7, pp.177661-177674.

[23] Saravanan A., & Venugopal, V. . (2023). Detection and Verification of Cloned Profiles in Online Social Networks Using MapReduce Based Clustering and Classification. International Journal of Intelligent Systems and Applications in Engineering, 11(1), 195–207. Retrieved from https://ijisae.org/index.php/IJISAE/article/view/2459.

[24] Shepperd, M., Hall, T. and Bowes, D., 2017. Authors' Reply to "Comments on 'Researcher Bias: The Use of Machine Learning in Software Defect Prediction'". IEEE Transactions on Software Engineering, 44(11), pp.1129-1131.

[25] Chakraborty, T. and Chakraborty, A.K., 2020. Hellinger net: A hybrid imbalance learning model to improve software defect prediction. IEEE Transactions on Reliability, 70(2), pp.481-494.s

[26] Song, Q., Jia, Z., Shepperd, M., Ying, S. and Liu, J., 2010. A general software defect-proneness prediction framework. IEEE transactions on software engineering, 37(3), pp.356-370.

[27] Artificial Intelligence and Software Engineering: Status and Future Trends, JörgRech, Klaus-Dieter Althoff, 2004

[28] The Role of Artificial Intelligence in Software Engineering, Mark Harman CREST Centre, University College London, Malet Place, London, WC1E 6BT, UK.

[29] Software Analytics: What's Next? Tim Menzies, North Carolina State University, Thomas Zimmermann, Microsoft Research. IEEE computer society 2018.

[30] The Synergy of Human and Artificial Intelligence in Software Engineering. Tao Xie, North Carolina State University, Raleigh, NC, USA. IEEE RAISE 2013.

[31] Towards supporting Software Engineering using Deep Learning: A case of Software RequirementsClassification. Ra´ul Navarro-Almanza, Reyes Ju´arez-Ram´ırez, Guillermo Licea, School of Chemical Science and Engineering, Universidad Aut´onoma de Baja California Tijuana, Baja California, M´exico. Calzada Universidad 14418. 2017 5th International Conference in Software Engineering Research and Innovation (CONISOFT).

[32] Machine Learning for Software Engineering: Models, Methods, and Applications. Karl Meinke, AmelBennaceur.

[33] Empirical comparison of machine learning algorithms for bug prediction in open source software, Ruchika Malhotra ; LaavanyeBahl ; Sushant Sehgal ; Pragati Priya, 2017 IEEE International Conference on Big Data Analytics and Computational Intelligence.

[34] Open Source Platforms and Frameworks for Artificial Intelligence and Machine Learning. Sambit Bhattacharya, Rajeev Agrawal, ErdemErdemir, BalakrishnaGokaraju, USA, IEEE

[35] Applying machine learning to predict software fault proneness using change metrics, static code metrics, and a combination of them. Yasser Ali Alshehri, Katerina Goseva-Popstojanova, Dale G. Dzielski and Thomas Devine. IEEE Conference SouthEastCon 2018.

[36] Surana, C.S.R.K., Gupta, D.B. and Shankar, S.P., 2019, May. Intelligent chatbot for requirements elicitation and classification. In 2019 4th International Conference on Recent Trends on Electronics, Information, Communication & Technology (RTEICT) (pp. 866-870). IEEE