Journal of Data Acquisition and Processing
ISSN: 1004-9037
https://sjcjycl.cn/
DOI: 10.5281/zenodo.98549627

AN EXPLORATION OF ARTIFICIAL INTELLIGENCE IN SOFTWARE DEVELOPMENT PROCESS

Dipali B. Tawar and Dr. Deepika Pathak
Department of Computer Application
Dr. A. P. J. Abdul Kalam University, Indore (M. P.) – 452010
Corresponding Author Email : dipali.tawar@gmail.com

Abstract
Developing a good software in terms of quality is a very difficult task. As it requires various phases of SDLC (Software Development Life Cycle) to achieve the task. Even though, by following the phases of SDLC the quality of software did not come up with the expected results. Various methodologies come into existence to conquer these difficulties and to foster superior grade, savvy and solid frameworks, different development approaches have been proposed. Two of these methodologies are Component-Based Software Engineering and Model-Driven Software Development. These two unique methodologies both endeavour to deliver superior grade, minimal expense and on time software frameworks with an attention on reusability and efficiency. In any case, there are huge issues identified with the utilization of models and segments during the development of software frameworks. To cope up with this problem, now a days in software development world, Artificial Intelligence (AI) is quickly gaining ground as a must-have technology. One of its major uses is to help automate manual processes for better efficiency of work processes, which is a game changer for any field that implements it. Artificial Intelligence inSoftware Engineering makes the tedious task of software development in easier way. Byadapting various advanced techniques available in AI, it makes the task easy and enhance the quality of software.

Keywords: Artificial Intelligence, Software Engineering, AI, Software Development Lifecycle, SDLC.

1. Introduction

Software Development Life Cycle is characterized as a deliberate methodology utilized by the software business to configuration, create, and test excellent software. The primary objective behind SDLC is to create top notch software that meets or surpasses client assumptions, arrives at consummation inside occasions and quotes.

SDLC comprises of the following phases:

- **Planning:** The most essential part of software development, necessity investigation is finished by the most talented and experienced software engineers in the association. Subsequent to get-together and investigating prerequisites from the customer, the degree report is created.
- **Implementation:** Here, the software engineers begin composing the code according to the analyzed prerequisites
- **Testing:** It is a vital stage which plans to find any mistake, bugs, or defects in the software

• **Documentation:** Each action acted in the venture is archived for future reference and upgrade in the development cycle.
• **Deployment and support:** The software is sent whenever it is endorsed for discharge.
• **Maintaining:** This stage happens once the product is operational. It includes adjustment of few highlights throughout some time. This stage likewise centres around observing the framework performance, bug amendment, and carrying out mentioned changes.

As indicated by IEEE software development measure is a cycle by which client needs are converted into a software product. The cycle includes making an interpretation of client needs into software prerequisites, changing the software necessities into configuration, executing the plan in code, testing the code, and here and there, introducing and looking at the software for operational use. The fundamental target of the development of a framework is its effective incorporation, all things considered, circumstances. Different software development techniques have been received to foster the software products, for example, cascade model, iterative and steady model, winding model, V model, fast application development, prototyping model, coordinated model, and crossover twisting model. The absolute most normally utilized are cascade, twisting, V model, and dexterous model.

There are different issues in conventional software development measure. The disappointment of numerous software projects as far as not gathering client/business prerequisites, inclined to mistakes and so forth has prompted software quality getting one of the main points of interest from all stakeholders‟ viewpoint. In a cutthroat climate quality-based product is fundamental requirement for any product achievement. To accomplish quality, effective interaction is required.

Software configuration is following stage to determine quality to make total construction or engineering of software which is expressed into necessity detail. Configuration gives not exclusively to discover how the software product will be show up, yet additionally permits both software clients and designers to acknowledge how it will work. Since configuration is the best way to totally make an interpretation of a prerequisites into a completed product. After software

plan software coding/performance stage is utilized for carrying out the software. Software performance depends on programming language. This stage likewise assume a significant part since utilizing coding an executable form of software is made. Programming language can impact the coding interaction, yet additionally the properties of the subsequent product and its quality. Software testing is directed when executable software exists. Testing used to discover mistakes and fix them to help software quality. Testing check what all capacities software expected to do and likewise watch that software isn't doing what he shouldn't do.

In contrast with numerous different cycles, the plan and development measure is particularly difficult to explore and oversee. Analysts have fostered various cycle models to comprehend, improve, and support the thinking about its specific qualities. In any case, the intricacy is with the end goal that no single model can resolve every one of the issues. Besides, the numerous models that have been created are different in concentration and detailing.

Specifically, the will in general include critical components of curiosity, intricacy, and cycle. The accompanying sections present these interrelated issues and diagram how interaction models can assist with tending to them, prior to proceeding onward to examine this current article's commitment. This development model depends on the possibility that when dangers are conspicuous, the judicious approach is by a progression of emphases where each cycle includes doing exercises that investigate the difficult space and additionally foster likely arrangements in more detail. Every emphasis is finished by a control cycle where the allure of the following cycle is thought of. Every cycle is nittier grittier and includes a greater responsibility of assets, however ought to have an expanded likelihood of an effective in general result. The burden of the model is that when putting such a lot of time in the beginning phases, any issues emerging later on are frequently more troublesome and costly to reconsider. Furthermore, the more extended the cycles are, the more noteworthy the possibility that necessities characterized from the get-go in the process may presently don't be significant when the developers really start coding. In a cutthroat market, time to market can mean the distinction between a fruitful product and an product that is obsolete before it is delivered. At the point when this occurs, groups regularly wind up scrambling to modify and add highlights, making a cascading type of influence on the UI text. As the development group modifies the code, the content previously planned and composed by and large gets obsolete too, and you'll need to revaluate and revise data that may have effectively been executed and supported. In light of these downsides, software development groups started exchanging over to an iterative or steady way to deal with software development.

Since by following the various software development models few aspects didn’t get completed and development of software doesn’t come up with expected results. As few models require certain time in gathering requirements while some other models didn’t make up a good quality of software. So, to avoid these problems Iterative development model comes into existence like Agile Development.

Agile Development refers to a software development approach based on iterative development. Agile Software Development is an iterative and incremental approach to software development that emphasizes the importance of delivering a working product quickly and frequently. It involves close collaboration between the development team and the customer to ensure that the product meets their needs and expectations.Each iteration is considered as a

short time "frame" in the Agile process model, which typically lasts from one to four weeks. The division of the entire project into smaller parts helps to minimize the project risk and to reduce the overall project delivery time requirements. Each iteration involves a team working through a full software development life cycle including planning, requirements analysis, design, coding, and testing before a working product is demonstrated to the client. To develop a good quality of software Agile Development follows the phases which are as follows:

1. Requirements gathering: In this phase, you must define the requirements. You should explain business opportunities and plan the time and effort needed to build the project. Based on this information, you can evaluate technical and economic feasibility.

2. Design the requirements: When you have identified the project, work with stakeholders to define requirements. You can use the user flow diagram or the high-level UML diagram to show the work of new features and show how it will apply to your existing system.

3. Construction/ iteration: When the team defines the requirements, the work begins. Designers and developers start working on their project, which aims to deploy a working product. The product will undergo various stages of improvement, so it includes simple, minimal functionality.

4. Testing: In this phase, the Quality Assurance team examines the product's performance and looks for the bug.

5. Deployment: In this phase, the team issues a product for the user's work environment.

6. Feedback: After releasing the product, the last step is feedback. In this, the team receives feedback about the product and works through the feedback.

The simultaneous development model depends on the thought that simultaneousness exists across all framework or software development exercises in an undertaking. For instance, albeit a venture might be in its last stage, there are still staff included at the same time in exercises, like planning, coding, and mix testing, normally connected with singular phases of development. Exercises, like examination or configuration, exist simultaneously in various states or methods of conduct, for example, the "anticipating changes" state or the "done" state. At the point when it is required, the model characterizes a progression of occasions that will travel exercises from one state into another. The model can be applied as a worldview for the development of customer/worker applications. In a customer/worker framework the framework measurement incorporates the exercises of plan, gathering, and use and the segment measurement incorporates the exercises of plan and acknowledgment.

Framework and part exercises happen at the same time and the modules of the framework can be planned and grown simultaneously. The disorder life cycle is an augmentation of the simultaneous model. This model proposes that software development is a client designer innovation continuum. Every one of the fundamental periods of framework development are applied recursively to client needs and specialized detail of the software. The simultaneous and disarray models give a more practical perspective on framework development, as they don't bind frameworks and software development in a direct successive arrangement of steps. This gradual methodology is especially appropriate for web applications where highlights can be refreshed on a continuous premise, and clients may not know about inconspicuous enhancements and changes happening from one delivery to another.

Regular updates are extensively harder for bundled applications. Be that as it may, at specific occasions, new highlights and updates can be downloaded from the product site or from an executable document sent by means of email. Lithe development has negative aspects too. The primary arrival of the application might be inadequate in highlights, and clients evaluating the application in its underlying stages might be baffled by an absence of usefulness and by lacklustre showing. The content and direction you incorporate into the UI will go far toward getting clients amped up for utilizing the application. Moreover, on the grounds that the application is created on a component by-highlight premise, and regularly by various designers, it might needcohesiveness.

Software efficiency is a misleadingly straightforward idea, yet a question of some discussion. In spite of the fact that its soonest estimation was in lines of code each worker hours worked, a superior definition is the proportion between the practical worth of software delivered to the work and cost of creating it. There are a few different ways to gauge software efficiency, including Function Point Analysis, Cost Component Modelling, Cyclomatic Complexity, and program performance measurements that consider the expenses of running and keeping up the software. Utilizing these apparatuses, the software development interaction can be overseen and efficiency improved by reusing code to use existing projects, limiting revamp through dependability drives, and embracing sound development practices and principles.

Be that as it may, in any event, when sound practices are clung to, software efficiency may not expand as a result of conditions outside the control of the development group. This incorporates quickly changing developments and the fixed-cost conduct of huge pieces of the software development measure. The idea of software efficiency is definitely not a hypothetical theoretical. It is a basic piece of the software designing interaction. Understanding software usefulness gets significant in frameworks examination when you consider that great frameworks investigation improves software efficiency and software usefulness is a triumph proportion of frameworks examination. In standard financial terms, usefulness is the proportion between the measure of products or administrations created and the work or cost that goes into delivering them. The presumption that follows, then, at that point, is that software efficiency is the proportion between the measure of software delivered to the work and cost of creating it.

This is a basic hypothesis that seems, by all accounts, to be sensible, yet by and by turns into a question of some discussion. To characterize software efficiency, we should initially build up a meaning of software. At its most principal level, software is a PC program contained lines of code. Be that as it may, lines of code, all by themselves, are not the essential expectations of a software undertaking and clients frequently don't have a clue the number of lines of code are in the software they are purchasing.

Artificial intelligence (AI) is a vast range branch of computer science concerned with building smart machines capable of performing tasks that typically require human intelligence. It is used to make making intelligent machines, especially intelligent computer programs. It is related to the similar task of using computers to understand human intelligence. Artificial intelligence allows machines to model, and even improve upon, the capabilities of the human mind. It has ability to do tasks that are usually done by human being. Ability of AI includes the ability to make decisions. Under defined constraints and criteria, the machine can make the most suitable and accurate decision. Different regression models can help predict the estimated time frame

and cost estimation for the current project when trained with past project timelines and cost estimations. The supervised and unsupervised learning algorithms can significantly help a developer with advanced services.
The scope of AI in Software Development is huge, and the list goes on with other applications such as automated coding, automated UI design, etc.

2. Literature Review

Laszczyk et al., (2019) depicted a scientific classification based looking over 38 of the current exhibition measurements and their definitions alongside their benefits and disservices. They guarantee their proposed reciprocal arrangement of measurements can make significant outcomes when utilized on arrangement sets got by solvers. It is another investigation of performance measurements as of late distributed, which planned to zero in on utilizing measurements gathered into four classifications: Cardinality, intermingling, appropriation, and spread. The examination hole detailed in this paper is the requirement for new measurements that can handle the limits looked by the HV. Be that as it may, present technique didn't yield subjective and quantitative based danger control factors during assessment. To control the dangers on programming projects with further developed procedures and to check the adequacy, another system is executed called Bayesian Statistics Software Process based Risk Control. Bayesian progressed device gauges the strategy subjectively and quantitatively on multivariate programming's and furthermore deals with the parametric factors, for example, mean score for hazard control effectiveness, hazard control time and normal run length on joint programming checking.
Anne Martens et al (2014), Quantitative expectation of quality properties (for example extra useful properties like execution, dependability, and cost) of software structures during design upholds a precise software designing methodology. Designing structures that display a decent compromise between different quality rules is hard, on the grounds that even after a utilitarian design has been made, many excess levels of opportunity in the software engineering range a huge, spasmodic design space. In current practice, software engineers attempt to discover arrangements physically, which is tedious, can be mistake inclined and can prompt imperfect designs. We propose a mechanized way to deal with search the design space for great arrangements. Beginning with a given introductory structural model, the methodology iteratively adjusts and evaluates engineering models. Our methodology applies a multi-rules hereditary calculation to software designs modelled with the Palladio Component Model. It upholds quantitative execution, dependability, and cost forecast and can be reached out to other quantitative quality measures of software models. We validate the immaterialness of our methodology by applying it to an engineering model of a segment-based business data framework and examine its quality measures compromises via naturally investigating in excess of 1200 alternative design competitors.

3. Research Methodology

As most of the software development follows the various models and also Agile Development but the software quality, productivity and output are not satisfactory.

In the past few years, businesses worldwide have slowly been upgrading their systems and processes with the latest in artificial intelligence, machine learning, and data science. However, the completely unanticipated yet deadly arrival of the COVID-19 pandemic and the ensuing global recession has convinced many businesses of the benefits of AI and related technologies.

Also in software development world, artificial intelligence is quickly gaining ground as a must-have technology. One of its major uses is to help automate manual processes for better efficiency of work processes, which is a game changer for any field that implements it.

According to a study conducted by the MIT Sloan Management Review, approximately 58% of businesses were expecting to see a significant change in their business models by 2023 due to AI.

In this paper we are putting forward five different ways where AI and software development go hand in hand today.

AI and the Demand for Software Developers in the Future

One of the most notable advancements in this field is the development of language models such as ChatGPT. ChatGPT, developed by OpenAI, is a powerful and highly advanced language processing model that is capable of understanding and generating human-like text. It has been trained on a massive dataset of text, allowing it to understand and respond to a wide range of topics and tasks. This technology has the potential to revolutionize the way businesses interact with customers, automate repetitive tasks, and extract valuable insights from large amounts of data. As in software development with the help of AI task for developing a software becomes easy but the role of software developers did not change. In market Software development professionals in the field of AI, ML and deep learning, Big Data, and more, are steadily on the rise, and it is expected to rise even more in the coming years.

AI Helping Products Reduce Time-to-Market

As we mentioned earlier, low-code implementation platforms are becoming popular nowadays, especially among small businesses that require a simple, yet robust software product released to the market in the shortest time possible. Besides low code, however, there is another technology being used in the market, albeit in its infancy, called automatic code generation.

AI code generators can use the concept of NLP, or natural language processing, to create actual code out of pseudocode and natural language commands. And while the technology is underutilized in the industry currently, the development of better training datasets and improved processing algorithms are helping it become a formidable player in the industry.

Improved Data Security

Data has become a hot commodity in recent years, and some people and organizations are willing to exploit the holes in the system to breach it and use that data for malicious reasons.

Many mega corporations and tech giants have had data breaches in recent years, which has given rise to a new wave of data security measures being implemented.

One of those recent technologies is AI and ML, which are now being used to help developers identify vulnerabilities right at the codebase level so that the issue is addressed as soon as possible. Moreover, these tools are also able to identify actual vulnerabilities, while ignoring false alarms and red herrings, reducing the wasted time during testing.

Improved Costing Estimates for Projects

Whenever software development project is in process one of the biggest and most time-consuming tasks is cost estimation. The reason for that is that the process of estimating the cost of a project relies on many factors, both external and internal.

Elements like project duration, complexity, and many other project parameters can affect those estimates. Therefore, it often requires that a team of experienced professionals sit together to come up with this estimate. It is not possible for individual to calculate the cost estimation for project. The solution is using predictive analytics and historical data from previous projects, both successful and failed, AI can help you come up with a good estimate for project. And with that cost estimate in hand, company decision-makers can decide what to do, such as finding the best CMS for their website that fulfills their needs and fits within the budget.

Helping Stakeholders Make the Right Decisions

AI in software development help in charge make sense of the data in front of them, and make the right decisions by using the insights AI generates from that data.

Constant evaluation of the data gathered during the project development phase to make smart decisions is one of the most popular agile team characteristics in demand today. However, with the huge amounts of data being created today, it is difficult for someone to wade through all that and generate the insights necessary to make the right decisions.

It can be possible by using artificial intelligence which help to analyze thedata to look for trends or anomalies within it, highlighting what should and shouldn’t continue doing.

4. Conclusion

Concluding, this paper proposed a latest technology – AI (Artificial Intelligence) by using it we can enhance the software development productivity. Artificial Intelligence has started to impact the business processes of many industries. Its actual potential extends far ahead of what we are witnessing today. And combined with other modern technologies like big data, ML, cloud, and Blockchain, it will soon change the way we see and approach software development.Software developers and software companies should adopt the opportunities available in AI
field in order to enhance the productivity of software development. Software developers should update their knowledge and try to inculcate the new emerging trends of AI in their daily work

practice related to Software engineering in order to increase the software developmentproduction in better way and to work hassle free.

REFERENCES

1. S.J. Chen, "Inverted A-Model for Stable Software Development, Journal of Software", Vol. 37, issue 12, Page 07-11, 2016.
2. R. P. Buse and W. R. Weimer, "Learning a metric for code readability," IEEE Transactions on Software Engineering, Vol. 36, Issue 4, Page 546-558,2010
3. Umakant Dinkar Butkar, et.al "Accident Detection and Alert System (Current Location) Using Global Positioning System" JOURNAL OF ALGEBRAIC STATISTICSVol. 13 No. 3 (2022) e-ISSN: 1309-3452
4. Mr. Umakant Dinkar Butkar, Manisha J Waghmare. (2023). An Intelligent System Design for Emotion Recognition and Rectification Using Machine Learning. *Computer Integrated Manufacturing Systems*, *29*(2), 32–42. Retrieved from http://cims-journal.com/index.php/CN/article/view/783 .
5. Mr. Umakant Dinkar Butkar, Manisha J Waghmare. (2023). Hybrid Serial-Parallel Linkage Based six degrees of freedom Advanced robotic manipulator. *Computer Integrated Manufacturing Systems*, *29*(2), 70–82. Retrieved from http://cims-journal.com/index.php/CN/article/view/786
6. Mr. Umakant Dinkar Butkar, Manisha J Waghmare. (2023). Novel Energy Storage Material and Topologies of Computerized Controller. *Computer Integrated Manufacturing Systems*, *29*(2), 83–95. Retrieved from http://cims-journal.com/index.php/CN/article/view/787
7. Mr. Umakant Dinkar Butkar, Manisha J Waghmare. (2023). Advanced robotic manipulator renewable energy and smart applications. *Computer Integrated Manufacturing Systems*, *29*(2), 19–31. Retrieved from http://cims-journal.com/index.php/CN/article/view/782
8. Mr. Umakant Dinkar Butkar, Manisha J Waghmare. (2023). Crime Risk Forecasting using Cyber Security and Artificial Intelligent. *Computer Integrated Manufacturing Systems*, *29*(2), 43–57. Retrieved from http://cims-journal.com/index.php/CN/article/view/784
9. Mr. Umakant Dinkar Butkar, Dr. Pradip Suresh Mane, Dr Kumar P K, Dr. Arun Saxena, Dr. Mangesh Salunke. (2023). Modelling and Simulation of symmetric planar manipulator Using Hybrid Integrated Manufacturing. *Computer Integrated Manufacturing Systems*, *29*(1), 464–476. Retrieved from http://cims-journal.com/index.php/CN/article/view/771
10. Laszczyk, M.; Myszkowski, P.B. Survey of quality measures for multi-objective optimization. Construction of complemen-tary set of multi-objective quality measures. Swarm Evol. Comput. Issue 48, 109–133,2019
11. Shirdastian H, Laroche M, Richard M-O. Using big data analytics to study brand authenticity sentiments: the case of Starbucks on twitter. Int J Inf Manag. Vol.48, :Page 291–307,2019.

12. Abdelghafar M. Elhady, Ahmed Abou Elfetouh S. &Hazem M. El-bakry, A. E. Hassan, "Generic Software Risk Management Framework for SCADA System", International Journal of Computer Applications (0975 – 8887) Vol 70, Issue 3, 2013.
13. Paul L. Bannerman., "Risk and risk management in software projects: A reassessment," The Journal of Systems and Software Vol.81, Page 2118–2133,2008.
14. Prabhakar Rao & P Seetharamaiah, "Organizational Strategies and Social Interaction Influence in Software Development Effort Estimation", IOSR Journal of Computer Engineering, Vol 16, Issue 2, 2014.
15. Murthi, S., "Preventive Risk Management for Software Projects", IEEE IT Professional Vol: 4, Issue. 5, Page 9 – 15,2002.
16. Sandra Miranda Neves, Carlos Eduardo Sanches da Silva, Valério Antonio Pamplona Salomon, Aneirson Francisco da Silva, Bárbara Elizabeth Pereira Sotomonte, "Risk management in software projects through Knowledge Management techniques: Cases in Brazilian Incubated Technology-Based Firms", International Journal of Project Management Vol 32, Issue 1, Page 125-138, 2014.
17. Shay Artzi, Julian Dolby, Frank Tip and Marco Pistoia, "Fault Localization for Dynamic Web Applications", IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, Vol. 38, Issue. 2, Page 314 – 335, 2012.
18. S. Saxena and S. K. Dubey, "Impact of Software Design Aspects on Usability," International Journal of Computer Applications, Vol. 61, Issue 22, Page 48-53, 2013.
19. Gaurav Kumar, Pradeep Kumar Bhatia," Impact of Agile Methodology on Software Development Process"," ISSN 2249- 6343International Journal of Computer Technology and Electronics Engineering (IJCTEE)Vol. 2, Issue 4, 2012.
20. H. Zhu and I. Bayley, "On the Composability of Design Patterns," IEEE Transactions on Software Engineering, Vol. 41, Issue 11, Page 1138-1152, 2015.
21. B. Singh and S. Gautam, "Hybrid Spiral Model to Improve Software Quality Using Knowledge Management," International Journal of Performability Engineering, Vol. 12, Issue 4, Page 341-352, 2016
22. S. Ali, L. C. Briand, H. Hemmati, and R. K. Panesar-Walawege, "A systematic review of theapplication and empirical investigation of search-based test-case generation," IEEE Transactions onSoftware Engineering, Page 742–762, 2010.
23. Coulin, C., Zowghi, D., & Sahraoui, A. MUSTER: A Situational Tool for RequirementsElicitation. In F. Meziane, & S. Vadera (Eds.), Artificial Intelligence Applications for ImprovedSoftware Engineering Development: New Prospects Page 146-165,2010.
24. M. Harman and P. McMinn, "A theoretical and empirical study of search-based testing: Local,global and hybrid search," IEEE Transactions on Software Engineering, Vol. 36, Issue 2, Page 226–247,2010
25. S. M. Poulding and J. A. Clark, "Efficient software verification: statistical testing using
26. automated search," IEEE Transactions oSoftware Engineering, Vol. 36, Issue. 6, Page. 763–777, 2010.

27. Suhaimi Ibrahim, NorbikBashah Idris, Malcolm Munro, and Aziz Deraman, "Integrating Software Traceability for Change Impact Analysis", The International Arab Journal of Information Technology, Vol. 2, Issue. 4, Page 301-308,2005.