Vol-08 Issue 01, January- 2024 ISSN: 2456-9348
Impact Factor: 6.736

IJETRM
International Journal of Engineering Technology Research & Management
www.ijetrm.com

DESIGN, DOCUMENTATION AND VALIDATION OF THE SOFTWARE ENGINEERING BY VERIFY THE SOFTWARE DESIGN AGAINST SRS USING ARTIFICIAL INTELLIGENCE TECHNIQUE

Jyoti Sisodia[1]
Dr. Suraj V Pote[2]
[1]Research Scholar, Department of Computer Science & Engineering, School of Engineering and TechnologyUniversity of Technology, Jaipur
[2] Supervisor, University of Technology, Jaipur

Abstract

Software engineering refers to the standard used to develop software. Software development is a human-centric activity that is time consuming and extremely complex. The concept of artificial intelligence and how to use its various techniques in software engineering (such as) B. How is artificial intelligence technology related to software engineering? This white paper describes many artificial intelligence techniques that can be used to tackle various software development tasks that can extend the nature of the program. Identifying, validating, and testing software requirements and designs areimportant tasks in the software development process and must be taken seriously.

Software activity has the advantages of lower maintenance costs, consistent software quality, and more customer-friendly software by investing direct efforts in these efforts. Still, many people working on these projects find that the methods available are very difficult, out of the ordinary, or simply unsuitable for the task. Artificial intelligence (AI) has become a common term for self-regulating IT applications, but its importance in software engineering has received little attention.

This study combines a detailed study of previous studies in this area with five subjective meetings with software designers who use or need to use AI devices in their daily work, and the current state of development, Evaluate future development potential and the risksof AI applications. Software engineering. In the software development life cycle, validationorganizes information

Keywords:
Software Engineering, Software Design, Artificial Intelligence Techniques.

INTRODUCTION

In terms of the number of utilitarian and nonfunctional needs that software seriousframeworks must support, the frameworks we promote these days are becoming increasingly mind-boggling. The impact of poor quality on the mission of these frameworks in a variety of basic applications can be disastrous. Furthermore, the cost of software development determines the total cost of such frameworks.

Over the last two decades, research into the application of artificial intelligence technologyto software development has exploded, resulting in a huge number of projects and distributions. Various gatherings and diaries have been established to disseminate the results of the examination in this field. It is advocated that AI techniques be used to reducethe chance for software frameworks to be marketed and to improve the nature of softwareframeworks. However, the examination local region continues to use a significant numberof these AI techniques, with minimal effect on the cycles and instruments used by the practicing software developer.

Artificial intelligence has become a popular term in both popular and academic circles. Thefollowing are some of the most noteworthy predictions and cutting-edge mythology relatedwith artificial intelligence: In the most pessimistic possibility, computers would take overold-style human engineering and improvement jobs, attempt to entirely replace human efficiency through cunning robotization, and oversee a machine-ruled fascinating modernlifestyle. In such circumstances, traditional software architects may become obsolete as machines take control of their projects.

Artificial intelligence is now a group of PC-basedprogrammers who mimic human intelligence in ways to achieve new goals throughdecision making, reviewing new data and integrating it into existing information structures,and using subjective or quantitative data. It is a general term for. Probabilistic evaluation.

Engineers acquire the project's business and specialized requirements at the necessity gathering stage. The design and work of the not fixed in stone in the design stage based onthe needs received. The solution to the problem is conceived and implemented during the

code development stage. The testing step is where analysts run various tests to ensure thateverything goes as planned. The arrangement is sent to the client during the sending stage.Finally, in the support stage, issues are addressed if necessary or discovered.

1. Software Requirements Specification

Software Requirements Specifications (SRS) is a diagram of future software frameworks.It is created according to a set of business requirements (CONOPS). The software requirements specification contains a set of use cases that represent both functional and non-functional requirements and the client connections that the software must provide to complete the communication.

The basis of the agreement between the client and the temporary worker or supplier on howthe software works is defined in the software requirements specification (in market- oriented projects, these tasks are taken over by the promotion and improvement departments. Will be). Before a clearer framework design is organized, the definition of software requirements is a thorough examination of the requirements with the aim of reducing subsequent redesigns. It should also serve as a useful basis for assessing the cost,risk, and deadlines of an item. When used correctly, software requirements specifications help prevent software project failures.

The software requirements specification report contains all the requirements that are important and necessary for your business to grow. To derive requirements, engineers needto have accurate knowledge of the product they are working on. This is achieved by maintaining continuous and detailed communication with workgroups and customers throughout the software development process.

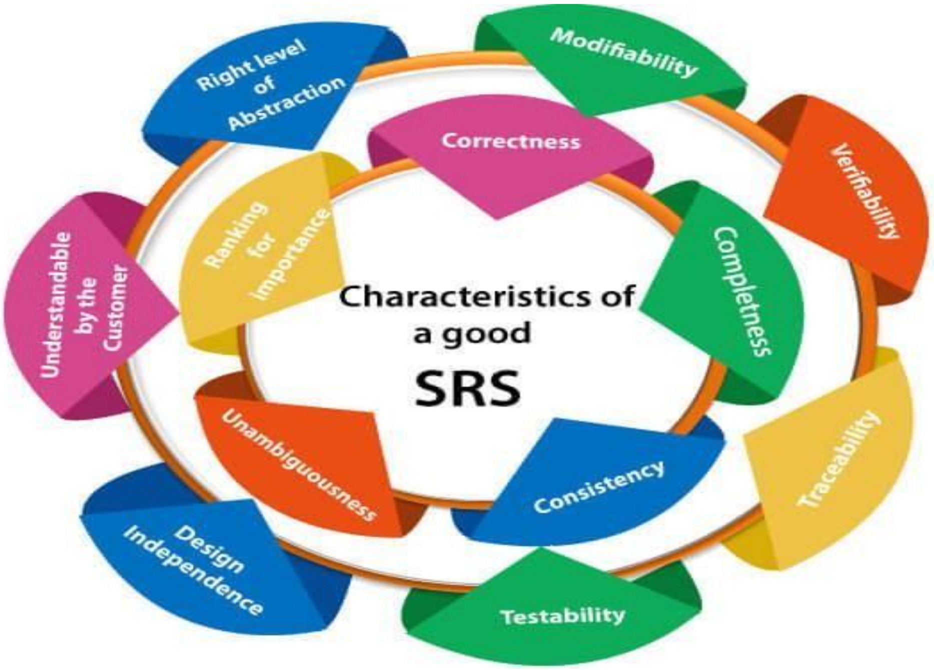

Figure: 1. Characteristics of good SRS.

1.1. Software Requirement Specification (SRS) Format

As the name implies, successful software framework development requires complete specifications and statements of software requirements. Depending on the type of prerequisite, these criteria might be both practical and non-utilitarian. The link between diverse clients and project workers is made since it is critical to fully understand the needsof clients. Based on the information gathered through collaboration, SRS is developed, which depicts software requirements that may include alterations and adjustments that must be made in order to maintain the item's nature and satisfy the needs of the client.

1. Introduction
   i) Purpose of this document
   ii) Scope of this document
   iii) Overview
   2. Overview
   3. Functional requirements
   4. Interface requirements
   5. Performance requirements
   6.6. Design constraints
   7. Non-functional attributes
   8. Provisional schedule and budget
   9. Appendix

1.2. Software requirements specification vs software design specification

According to Ward and Mellor, there are several advantages to separating SRS and SDS. SRS and SDS can be confused because software requirements and design processes are often not performed independently and there is no clear consensus as to whether a particular aspect refers to SRS or SDS. SRSs involve design decisions. Poor SRSs can hinder design efficiency. SDSs often involve implementation decisions, causing the samedifficulty. Poorly written SDSs can hinder implementation. The SRS and SDS are difficult to identify.

An SRS describes what the software will perform but not how. An SRS should specifythe software's outcomes, not its methods. SRS needs to identify all software requirements,but not project management, design, implementation, or testing. Neither end users nor analysts are eligible to prepare their own SRS.

Software design specifications (SDS) transform software requirements specifications intothe software structures, components, interfaces, and data required for programming. It records design process findings and is used to communicate software design information. Designers create SDSs.

First, the system can be described in two ways. SRS describes a system in technical termsrather than computer hardware or software technology, so it applies regardless of the technology used to implement the system. SDS describes a system implemented by a particular technology. Then, by splitting SRS and SDS, you can split long processactivities into two smaller tasks: analyzing and defining software requirements and creating and defining software designs. Achieve SRS and SDS consistency.

2. Artificial Intelligence Techniques

Improvements to the master framework: - Expert frameworks manage the arranginginteraction using information rather than information. Information engineers create frameworks by eliciting knowledge from experts, coding that information in a logical structure, approving the information, and then designing a framework using a variety of tools.

The following are the primary stages of the expert system development process:

- Preparation
- Information gathering and analysis

- Conceptualizing knowledge
- Programming
- Validation of information
- Assessment of the system

Possibility appraisal, asset distribution, assignment staging, and booking requirements study are all part of the arranging step. Gathering information is the most important step inimproving ES. During this phase, information engineers work with area managers to protect, organize, and investigate ES space data. The goal of information investigation is to break down and organize the data. Obtainable during the information gathering stage. We move on to the information design step after the information evaluation is completed.We are nearing the end of the design stage, and we have knowledge definition, natty grittydesign, and a decision on how to treat information, as well as a decision on an improvementtool. Examine whether it supports your pre-determined system, internal reality structure, and simulated connecting point. In the Expert System Development Life Cycle, coding thisstage takes the least amount of time. Coding, designing experiments, commenting code, nurturing User's manual, and setup instructions are all included.

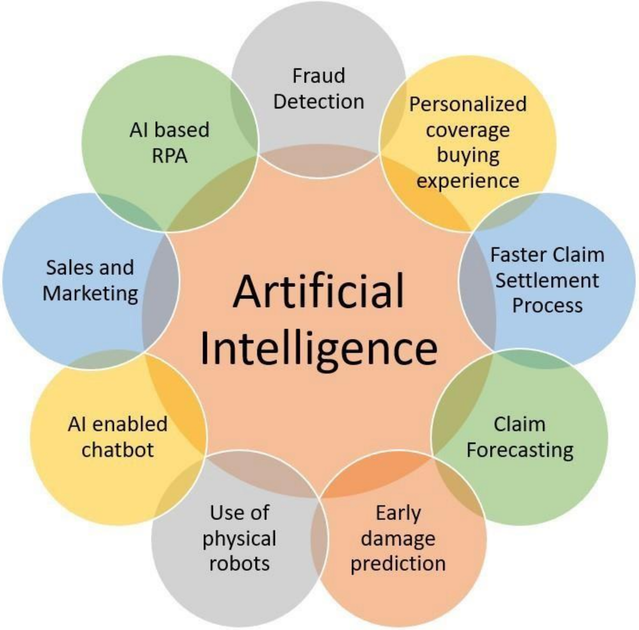

Figure: 2. The Artificial Intelligence Techniques.

2.1. Artificial Intelligence in Software Engineering

Software engineering is the application of formal engineering principles to the design anddevelopment of software. Software engineering is basically responsible for softwaredevelopment. Software development is a long process that involves multiple phases and requires the use of executable code. Humans have long written code to improve software, but no machine can beat it. AI refers to the process of creating intelligent machines that can perform human-like tasks.. Artificial intelligence techniques can be used to help with a wide range of software development tasks. As a result, there is significant opportunity for working on all phases of the SDLC (Software advancement life cycle).

3. Requirement Validation

The requirements approval phase is the final step in the requirements engineering process. The approval of requirements is completed to ensure that they are complete and predictable, as specified by the client. The software

iJETRM
International Journal of Engineering Technology Research & Management
www.ijetrm.com

requirements approval process identifies software requirements specification errors (SRS). During the requirements approval process, ambiguities and discrepancies in requirements are resolved [21].

These stages have a limited number of advances that verify the requirements; the methods are as follows:

- Checks for consistency
- Checks for culmination
- Checks for legality
- Verification of authenticity
- Checks for ambiguity
- Consistency.

These checks are carried out during the requirements approval stage to ensure the following:

- The requirements should be predictable with respect to one another, which meansthat no two requirements should compete or contradict one another.
- The criteria should be essentially achievable.
- The requirements must be fulfilled in every way.
- The requirements should include all relevant information.
- The requirements must address the framework's true requirements.
- The partners' criteria should be realistic.
- Each requirement should be presented in a way that prevents multiple interpretations.

➢ Challenges To Requirements Validation

Validating requirements using a method or framework is difficult. Due to a lack of skilledtechnical employees, training, or knowledge & skills, many firms do requirements validation "ad-hoc." Developers prioritise testing.

3.1. Requirement Validation Techniques

Using Requirement validation procedures ensures that user specifications are complete andthe SRS document is error-free. Industry practises requirements prototyping, reviews, viewpoint-oriented validation, and use-case based modelling. This paper discusses severalstrategies.

4.1.1. Inspections

Fagan 1976 introduced inspections as a way to detect problems. Inspections can discover 50-90% of flaws, according to research. Manual inspections verify work-products. A smallgroup of peers performs it to confirm it's correct and meets product specs. ISO/IEC 15504and CMMI suggest inspections for requirements validation.

4.1.2. Requirements Prototyping

Requirements Prototyping is a key tool for testing user needs because it represents a system's shell. Prototypes validate requirements by offering system knowledge. Prototypeshelp validate requirements when you're not sure you have a good set. Literature discussesthrow-away and evolutionary prototypes.

"Throwaway prototypes" identify misunderstood needs. After user feedback, throwaway prototypes are discarded after meeting initial requirements. The prototype's input helps thedevelopment team and clients resolve requirements issues. If both parties agree on the criteria, the prototype is discarded and the requirements are added to SRS.

"Evolutionary prototyping" is based on settled criteria and subject to software quality limitations. User input refines evolutionary prototypes based on original criteria.

4.1.3. Requirements Testing

Requirements testing validates the SRS, not the software system. Test cases are generatedfor all provided criteria, writing time, and/or economic resources. This expense is part of requirements validation. Requirements testing helps

identify confusing or incomplete requirements by indicating a problem with a requirement if a test case fails.
The test cases used to test the requirements could eventually be utilised to test the whole system. TCD inspections are test case based.
Tony Gorschek and Nina Fogelstrom proposed test-case-based software requirements inspection. This technique involves writing test cases to test system requirements and testing them.

4.1.4. Viewpoint-oriented Requirements Validation

Researchers have known for decades that more information sources improve knowledge. Different sources and witnesses may have different memories. This ensures the requirements' completeness and accuracy. To use this approach, compare and examine
different viewpoints systematically. This method aids elicitation. Viewpoint-orientedvalidation compares many views and resolves discrepancies.

4. The Role of AI Techniques in Software Development Activities

- ➢ **The Development Process**
- **Software requirements analysis**

i. Requirement Engineering (RE)

Within a group of documents, requirements are initially expressed in natural language. These documents are typically "the unresolved perspectives of a group of individuals and will, in most situations, be incomplete, inconsistent, conflicting, not prioritised, andfrequently overblown, beyond actual needs." This phase's major activities include requirement elicitation, collection, and analysis, as well as their transformation into a lessambiguous representation.
The following are some of the issues that have arisen during this phase:

- Unclear requirements
- Incomplete, unclear, imprecise requirements
- Incompatible requirements
- Volatile needs
- Inter-stakeholder communication is poor.
- Unmanageable requirements.

ii. Processing Natural Language Requirements NLR

The system is not implemented, but a framework for converting specifications written in NL (English) to formal specifications (TELL) has been introduced. The NL2ACTL systemwas introduced with the goal of translating NL sentences created to describe the functionality of reactive systems into action-based temporal logic statements. Another system, FORSEN, was created with the goal of translating NL requirements into the formalspecification language VDM. This method was able to detect the ambiguity of the NL requirement. A general methodology for automatically developing OO models from NL requirements using language tools was presented. NLOOPS, aimed at creating OOspecifications from NL requirements, was developed using the Large Object-Based Language Interactor Translator Analyzer (LOLITA) NLP system. An approach has been developed to connect the linguistic world with the conceptual world through a series of linguistic patterns. Another system, ClassModelBuilder (CMBuilder), was built as an NL-based CASE tool to create UML-defined class diagrams from NL requirements papers.

iii. Risk Management

Risk management processes are a means of predicting risks and implementing proceduresto prevent and / or mitigate the impact of those risks. The risk management process beginsin the analysis phase of the software development life cycle. However, the actual risk management process continues throughout the product development phase. With automatic programming techniques that make data structures adaptable, AI-based systems have no risk management strategy. Automatic programming is the development of programs on a computer, usually based on higher and easier-to-specify criteria than traditional programming languages. The goal is to make the specification smaller, easier to write, easier to understand (closer to the concept of an application), and less error prone than a programming language.

5. Conclusion

Artificial intelligence techniques that are used to automate necessity engineering exercisesare examined. The overall goal is It may be deduced that using AI techniques or AI calculations to necessity engineering is not a straightforward task. AI calculations, like necessity arrangements, necessitated pre-marked prepared data. However, it is commonly assumed that physically grouping requirements is a time-consuming task. Unlike traditional manual order procedures, AI techniques rely on the display of framework. For software necessity characterization, another Deep learning model, such as Intermittent Neural Network, can be used.

Software engineering aids us in the development of software products; yet, adhering to software engineering standards takes a long time. By incorporating AI approaches into thesoftware development process, the item's quality can be improved. We can kill risk appraisal carefully while saving time in software development and developing a successful product by using AI-based frameworks with the aid of robotized apparatus or computerisedprogramming apparatus. We can reduce the time it takes to improve software using Artificial Intelligence approaches in Software Engineering.

For software engineers who ignore the potential of AI in the software development life cycle and stick to routine tasks, automated schedules increase reliability and cost, and riskbeing replaced in the long run and losing their intended position. there is. To compete withartificial intelligence, software designers need to be more innovative and smarter in the future. A software development company that uses AI to identify software products fasterand more accurately, ignoring the acceptance of AI risks driven out of the market by moreinnovative competitors. Artificial intelligence is a fast-growing future innovation in software engineering, and early adopters take it seriously.

6. References

1. Acemoglu D, Restrepo P. Artificial intelligence, automation and work (no. w24196): National Bureau of Economic Research; 2018.
2. Dhar V. The future of artificial intelligence. Big Data Vol. 4 , N. 1; 2016.
3. Fetzer JH. Artificial intelligence: Its scope and limits (Vol. 4): Springer Science & Business Media; 2012.
4. Friedrich O, Racine E, Steinert S, Pömsl J, Jox RJ. An analysis of the impact of brain-computer interfaces on autonomy. Neuroethics. 2018:1–13.
5. Hameed Ullah Khan, Ikram Asghar, Shahbaz Ahmad AK Ghayyur, and Mohsin Raza. An empirical study of software requirements verification and validation techniques along their mitigation strategies. Asian Journal of Computer and Information Systems,3(3), 2015
6. Hany H Ammar, Walid Abdelmoe, and Mohamed Salah Hamdi,"Software Engineering Using Artificial Intelligence Techniques: Current State and Open Problems", 2010.
7. Helbing D, Frey BS, Gigerenzer G, Hafen E, Hagner M, Hofstetter Y, et al. Will democracy survive big data and artificial intelligence? In: Towards digitalenlightenment. Cham: Springer; 2019. p. 73–98.
8. Jeremy Dick, Elizabeth Hull, and Ken Jackson. Requirements engineering. Springer, 2017.
9. Kietzmann J, Pitt LF. Artificial intelligence and machine learning: what managers needto know. Bus Horiz. 2020;63(2):131–3.
10. Lu H, Li Y, Chen M, Kim H, Serikawa S. Brain intelligence: go beyond artificial intelligence. Mobile Netw Appl. 2018;23(2):368–75.
11. Makridakis S. The forthcoming artificial intelligence (AI) revolution: its impact on society and firms. Futures. 2017;90:46–60.
12. Masooma Yousuf and M Asger. Comparison of various requirements elicitation techniques. International Journal of Computer Applications, 116(4), 2015.
13. Niyaz Q, Sun W, Javaid AY. A deep learning based DDoS detection system in software-defined networking (SDN). arXiv preprint. arXiv. 2016;1611:07400.
14. Paola Spoletini and Alessio Ferrari. Requirements elicitation: a look at the future through the lenses of the past. In 2017 IEEE 25th International Requirements Engineering Conference (RE), pages 476–477. IEEE, 2017.
15. Sourour Maalem and Nacereddine Zarour. Challenge of validation in requirements engineering. Journal of Innovation in Digital Ecosystems, 3(1):15–21, 2016.

16. Sven Feja, Soren Witt, and Andreas Speck. Bam: A requirements validation and verification framework for business process models. In 2011 11th International Conference on Quality Software, pages 186–191. IEEE, 2011.
17. Syed Waqas Ali, Qazi Arbab Ahmed, and Imran Shafi. Process to enhance the quality of software requirement specification document. 2018 International Conference on Engineering and Emerging Technologies (ICEET), pages 1–7, 2018.
18. Wiegers, Karl; Beatty, Joy (2013). Software Requirements, Third Edition. Microsoft Press.
19. William Gryffyth StClair and Sumner Augustine StClair. Automated management of software requirements verification, February 3 2015.
20. YT Tiky. Software development life cycle. Hongkong: THe Hongkong University of Science and Technology, 2016.

Author's Declaration

I as an author of the above research paper/article, hereby, declare that the content of this paper is prepared by me and if any person having copyright issue or patent or anything otherwise related to the content, I shall always be legally responsible for any issue. For the reason of invisibility of my research paper on the website/amendments/updates, I have resubmitted my paper for publication on the same date. If any data or information given by me is not correct I shall always be legally responsible. With my whole responsibility legally and formally I have intimated the publisher (Publisher) that my paper has been checked by my guide(if any) or expert to make it sure that paper is technically right and there is no unaccepted plagiarismand the entire content is genuinely mine. If any issue arise related to Plagiarism / Guide Name / Educational Qualification/Designation/Address of my university/college/institution/ Structure or Formatting/ Resubmission / Submission /Copyright /Patent/Submission for any higherdegree or Job/ Primary Data/Secondary Data Issues, I will be solely/entirely responsible for any legal issues. I have been informed that the most of the data from the website is invisible or shuffled
or vanished from the database due to some technical fault or hacking and therefore the process ofresubmission is there for the scholars/students who finds trouble in getting their paper on the website. At the time of resubmission of my paper I take all the legal and formal responsibilities, If I hide or do not submit the copy of my original documents (Aadhar/Driving License/Any Identity Proof and Address Proof and Photo) in spite of demand from the publisher then my paper may be rejected or removed from the website anytime and may not be consider for verification. I accept the fact that as the content of this paper and the resubmission legal responsibilities and reasons are only mine then the Publisher (Airo International Journal/Airo National Research Journal) is never responsible. I also declare that if publisher finds any complication or error or anything hidden or implemented otherwise, my paper may be removed from the website or the watermark of remark/actuality may be mentioned on my paper. Even if anything is found illegal publisher may also take legal action against me

Jyoti Sisodia
Dr. Suraj V Pote

DECLARATION:

I AS AN AUTHOR OF THIS PAPER / ARTICLE, HEREBY DECLARE THAT THE
PAPER SUBMITTED BY ME FOR PUBLICATION IN THE JOURNAL IS COMPLETELY MY OWN GENUINE PAPER. IF ANY ISSUE REGARDING COPYRIGHT/PATENT/ OTHER REAL AUTHOR ARISES, THE PUBLISHER WILLNOT Be LEGALLYRESPONSIBLE. IF ANY OF SUCH MATTERS OCCUR PUBLISHER MAY REMOVE MY CONTENT FROM THE JOURNAL WEBSITE. FOR THE REASON OF CONTENT AMENDMENT/OR ANY TECHNICAL ISSUE WITH NO VISIBILITY ON WEBSITE/UPDATES, I HAVE RESUBMITTED THIS PAPER FOR THE PUBLICATION.FOR ANYPUBLICATION MATTERS OR ANY INFORMATION INTENTIONALLY HIDDEN BY ME OR OTHERWISE, I SHALL BE LEGALLY RESPONSIBLE. (COMPLETE DECLARATION OF THE AUTHOR AT THE LAST PAGE OF THISPAPER/ARTICLE