International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 9
Article Received: *25 July 2023* ***Revised:*** *12 September 2023* ***Accepted:*** *30 September 2023*

Exploring Software Development Change Analysis with an Emphasis on Requirements

Syed Adnan Afaq

Department of Computer Application, Integral University

Lucknow, India

Email: saafaq@iul.ac.in

Mohammad Faisal

Department of Computer Application, Integral University

Lucknow, India

Email: mdfaisal@iul.ac.in

Abstract- Most software requirements are not definitive of the development process. Rapid changes in user expectations, market conditions, and company practises all need regular updates to software. Requirement change management is a crucial and difficult component of every software development project. Project failure or cancellation often occurs because of requirements changes. Requirements Change refers to requirements that are added, removed, or amended during the system development life cycle. Requirements Change requires additional work in the design phase, which boosts the cost of developing the system, lengthens the time required, and reduces system quality. The paper investigates research efforts in the topic of requirement change and helps in determining the study's purpose. Various Requirement Change Management concepts and approaches are provided, and numerous activities are conducted to mitigate the effects of requirement changes. The study emphasizes on causes, attributes, prioritization of changed requirement, framework for RCM and Change Impact Analysis. This study briefly describes all the possible fact and figure about requirement change. The study includes various phases of requirement change management such as Requirement Elicitation, Requirement Change Identification in Requirement Document using Two Phase Requirement Document Comparison Algorithm, Prioritization of Changed Requirement using Fuzzy approach, Interdependency analysis and change impact analysis on various software project parameters such as time, cost and human resources.

Keywords- Change Impact Analysis; Prioritization; Requirement Change Causes; Requirement Change Management.

I. INTRODUCTION

In today's fast-paced and ever-changing business landscape, organizations face the challenge of responding to evolving market demands, technological advancements, and shifting stakeholder expectations. In this dynamic environment, managing changes to project or system requirements becomes paramount for ensuring successful outcomes. This is where requirement change management comes into play [1].

Requirement change management refers to the systematic process of identifying, evaluating, implementing, and tracking changes to project or system requirements throughout their lifecycle. It involves a structured approach to address modifications to the original set of requirements, enabling organizations to adapt and align their solutions with the changing environment.

The need for effective requirement changes management arises from several factors. Firstly, businesses operate in a world of constant change, driven by market trends, competitive pressures, and customer demands. Consequently, project requirements often need to be modified to meet emerging needs or seize new opportunities [2].

Secondly, stakeholders play a critical role in shaping requirements. Their evolving expectations, feedback, and inputs necessitate adjustments to project goals, features, or functionalities. By managing requirement changes, organizations can actively engage with stakeholders, foster collaboration, and deliver solutions that truly address their needs [3].

Moreover, technology advancements bring forth new possibilities and challenges. As innovative tools, platforms, and frameworks emerge, existing requirements may require modifications to leverage the potential benefits or mitigate

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 9
Article Received: *25 July 2023* ***Revised:*** *12 September 2023* ***Accepted:*** *30 September 2023*

risks. Requirement change management helps organizations stay abreast of technological advancements and harness them to their advantage [4].

Additionally, regulatory changes, industry standards, or legal obligations can impact project or system requirements. Organizations must adapt their solutions to remain compliant, adhere to industry best practices, and mitigate legal or operational risks. Requirement change management ensures that such changes are effectively incorporated, minimizing disruptions and maintaining compliance [5].

Implementing effective requirement change management involves a series of interrelated steps. It begins with the identification of proposed changes, followed by a comprehensive impact analysis to assess the implications on project scope, timeline, resources, and risks. Change evaluation involves prioritizing and evaluating proposed changes based on their importance, urgency, and alignment with project objectives [6].

Once a change is approved, it needs to be properly implemented, verified, and validated. Documentation and change tracking play a vital role in maintaining a clear record of all changes, their rationale, and their traceability.

II. LITERATURE REVIEW

Requirement change is an unavoidable state of the software development task. There are various model and framework for requirement change management which may mitigate the change. The impacts of change are hard to predict because of things like bad requirement specifications, the project's complexity, the team's lack of experience with handling changes, and the number of people involved, unanticipated actions, and biased estimation and decision making. Almost 70% of problems happen because clients' requirements keep changing during the requirement design and during the development phase, cultural diversity within development teams is responsible for 52% of problems, whereas strong communication is responsible for 88% of these problems, 70 percent are due to a lack of requirements management practises as suggested by [7] and 7 percent are due to other factors (Haleem & Farooqui, 2021a).

A. Summary of related work

Table 2: Brief summary of related work

Figure 1: Chaos Report outcomes 2015-2019

Table 1: Project Failure Factors

S. No.	Project Impaired Factors % of Responses	Percentages of Response
1	Partial Software Requirements	13.00%
2	Absence User Involvement	12.30%
3	Incomplete requirements	13.10%
4	Absence of Resources	10.50%
5	Impractical Anticipations	9.80%
6	Inappropriate Executive Support	9.40%
7	Changes in requirement	8.80%
8	Unstructured Planning	8.20%
9	Unmannered IT Management	6.20%
10	Technical Knowledge	4.31%
11	Other	9.90%

Most of the software projects are failed due to

several reasons like- Lack of Requirement Analysis, Poor Requirements Quality, Incomplete Requirement, Changing Requirements & Specifications. The organizations are classified by Standish Group according to their revenue such as

large, intermediate and small organization [8].

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 9
***Article Received:** 25 July 2023 **Revised:** 12 September 2023 **Accepted:** 30 September 2023*

S. No.	Author	Suggested Approach	Case Study	Attributes	Methodology
1	M.A. Akbar, 2019	RCM Framework	SRCMIMM implementation and software RCM development	Safety and Security	Stage 1: Establish Change Stage 2: SRCMIMM adoption
2	A. A. Alsanad, 2019	Multilevel ontology framework	Through the use of a survey and a case analysis	Understandability	SWOT evaluation of RCM methods
3	Saleem, 2019	Requirement change management model	Software Configuration Management	Consistency	Step1: Examine the causes of change Step 2: Prevent situations of change
4	Pai Zheng, 2019	Innovative strategy	Indefinite	Serviceability	Evaluation changes
5	Y. Yan, 2019	Knowledge management system (Knowledge transfer and sharing methods)	Indefinite	Security, Safety, Reliability, Efficiency	Design of KMS, factors affecting knowledge transfer and methods used to implement knowledge Transfer
6	M. A. Akbar, 2018	Requirement change analysis	Global Software Development GSD Paradigm	Efficiency Reliability	Step 1: Changeanalysis Step 2: ChangeIdentification Step 3: Handle challenges
7	S. Answer ,2018	CMP	Indefinite	Risk assessment, protection, safety, and trustworthiness	Change Analysis
8	M. Shafiq, 2018	RCM framework	Indefinite	Serviceability Consistency	For assessment, surveys and expert interviews are used.
9	K. Chari, 2017	The waterfall method	Unspecified	Serviceability Consistency	Step 1: Compare flaws Step 2: Inject flaws Step 3: Examine
10	S. Jayatilleke, 2017	RC Analysis	Course Management System	Comprehensibility Effectiveness, Consistency	Step 1: Change evaluation Step 2: Alter Identification Step 3: Determine the dependencies

III. CAUSES AND ATTRIBUTES OF REQUIREMENT CHANGE

A. *Causes of Requirement Change*

The majority of today's large companies utilise various tools and methods to handle the changes to their software as well as the requirements for new features. These processes are meant to govern any and all change requests, as well as the lifecycle of project development [9]. On the other hand, some of these tools are software tools, while others are lightweight tools (pen and paper). The majority of these procedures concentrate on recording change request, tracking requirements,

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 9
Article Received: *25 July 2023* ***Revised:*** *12 September 2023* ***Accepted:*** *30 September 2023*

requirement traceability, and managing change requests. However, the subsequent parts will be categorised and evaluated based on the techniques that are currently being used for change impact analysis.

The fact that requirements are in a state of change is one of the primary reasons why software is frequently updated. The requirements for the software can change at any time during the process, beginning with the elicitation phase and continuing all the way until the completion of the project [10]. Changes to requirements also signify how the system has to evolve in order to continue providing value to its users and to preserve its standing as a competitive product on the market. Modifications of this nature, on the other hand, pose a substantial threat since they can result in the degradation of the programmed system [11]. Therefore, modifications to requirements ought to be documented, monitored, and tightly regulated in order to assure the system's continuous existence from a technical perspective. This may be done by keeping a record of the changes and comparing them to the original requirements [12].

Changes in requirements are caused by many different things. Changes in requirements can be caused by both internal and external factors [13].

Figure 2: Causes of Requirement Changes

B. *Attributes of requirement change*

Requirement change refers to any modification made to the documented specifications, functionalities or features of a product, system or service during the development process or after it has been deployed [14]. The following are some attributes of requirement change:

Nature: Requirement changes can be minor, such as spelling corrections, or major, such as changes to the fundamental design or architecture of a system.

Frequency: Requirement changes can occur at any stage of the development lifecycle, from the planning phase to the deployment phase, and can occur frequently or infrequently.

Impact: Changes to requirements can have varying impacts on the project. Some changes may be simple to implement, while others may require significant effort and resources, potentially impacting the timeline and budget of the project.

Triggers: Requirement changes can be triggered by

various factors, such as new business needs, changes in regulations, or feedback from users.

Table 3: Requirement Change Factors

1. External Factors: Government regulations and Market competitors	
2. Internal Factors:	
1. Technical view point	a. Product limitations and constraints b. Deficiency in the experience d. Size of the software project e. Price of Software and hardware
2. New adaptations to the business and work setting	
3. Requirement changeability	
4. Requirement variety	
5. Analyzability of specifications	
6. User-project team miscommunication.	
7. Irresponsible behavior of clients	

Approval process: Most requirement changes require approval before they can be implemented. The approval process may involve stakeholders, project managers, developers, and quality assurance teams.

Communication: Communication is crucial during the requirement change process to ensure that all stakeholders are aware of the changes and their impact.

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 9
***Article Received:** 25 July 2023 **Revised:** 12 September 2023 **Accepted:** 30 September 2023*

Documentation: Changes to requirements should be documented thoroughly to ensure that all stakeholders are aware of the changes and can reference them in the future.

Testing: After a requirement change is implemented, it should be thoroughly tested to ensure that it does not negatively impact the system and that it meets the new requirements.

Traceability: The traceability of requirement changes is important to ensure that changes are properly implemented and tested, and to enable effective project management and future reference.

Table 4 precise various attributes and measures associated to Requirement Change available in the literature. These attributes are suitable in case of well-documented requirements. In contrast, because each organisation or stakeholder has its unique manner of interpreting or expressing requirements, these attributes must be adapted.

Table 4: summary of requirement change attributes

1.	Size of requirements Number of lines in requirement document Number of words in requirement document
2.	Change type
3.	Change request form
4.	Number of Total requirements
5.	Number of revised requirements
6.	Impact of requirement change
7.	Change frequency
8.	Priority of requirement change

From the above-mentioned measures, most of the attributes satisfy basic measurement
principles, four size measures are shown to be good assessors to determine number of changes.
In this study mainly two factors are focused which is based on size of the projects that is No. of lines and No. of words.

IV. REQUIREMENT CHANGE MANAGEMENT FRAMEWORK OUTLINE

Literature review and analysis reveals the impacts and consequences of Requirement Change on the successful development of a software product. Hence, there is a need for an effective framework to efficiently handle requirement changes, and this study introduces an efficient Requirement Change Management framework designed for early-stage integration within the SDLC. The proposed framework concentrates on the core issues associated with the emergence of change requirements and proposes an approach to avoid project failures. The framework as shown in Figure 3.1 consist of four major components: Input component in the form of Initial Requirements; Core Process as the combination of Change Identification Technique including Interdependency Analysis, Prioritization of Change Requirements, Change Impact Analysis techniques which is final output component in the form of Impact on Cost, Time and Human Resource.

The execution of the proposed framework begins with the elicitation of requirements from the stakeholders, taking as initial requirement as input at "t" time and the Initial Requirement Document is generated. After "t+d" time change request is generated and Revised Requirement Document is generated.

The next step involves identification of change requirements in the requirement document. At this phase requirement change is being identified by Two Phase Requirement Document Comparison Tool, in which lines of both requirement document are compared. After finding the changed lines in the requirement changes are identified.

Next phase involves the prioritization of change requirement. It initially involves analyzing the interdependencies among requirements by performing interdependency analysis that results in interdependency matrix. The prioritization is accomplished through fuzzy approach in which three member functions are taken. Interdependency Level, Difficulty Index and Change Rating. At last, change impact analysis is performed, where three parameters have been taken for analysing the change.

The outcome of this stage includes the impact of change on cost, time and human resources. The framework is finally validated by using finite state machine and Test cases are generated to support the validation of the framework.

A survey and analysis of the literature reveals the impacts and consequences of requirement change on the successful development of a software product. As a result, a capable framework is required to efficiently manage the requirement change.

The proposed framework focuses on the key difficulties related with the establishment of change requirements and suggests a strategy for avoiding project failures.

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 9
Article Received: *25 July 2023* ***Revised:*** *12 September 2023* ***Accepted:*** *30 September 2023*

The proposed framework consists four major phases for requirement change management. The initial phase involves the collection of requirements from stakeholders. Moving on, the second phase is dedicated to the identification of requirements for change. Subsequently, in the third phase, the emphasis is on prioritizing these changed requirements. Finally, the fourth and concluding phase entails a thorough analysis of the impact of changes, assessing their impacts across various parameters such as time, cost and human resources.

A. *Features of Proposed Framework*

- Identify and classify the significant concerns and problems of Requirement change and its impact on Software projects.
- Propose an approach for Identification of Requirement change in the Requirement Document at an early stage using Two Phase Requirement Document Comparison Tool.
- Construct an Algorithm for Identification of Requirement Change in Requirement Document.
- Perform interdependency analysis among various changeable requirements and compute the interdependency level using interdependency graph and matrix.
- Perform changed Requirement Prioritization using Fuzzy logic.
- Validate framework using finite state machine validation technique.

B. Proposed Framework: Requirement Change Management Framework

In consideration of the ever-evolving requirements of the software industry, academic societies used to lend a helping hand to the software industry by continuously developing more advanced tools, methods, and procedures. This was done in response to the ever-shifting demands of the software industry. Some of the largest software companies have their own research and development centres to ensure their software meets their needs as they change.

Through this research, it is hoped to give the software industry and the educational society a conceptual boost by coming up with a new way to find changeable requirements at the earliest stage of the SDLC and a way to deal with requirement change.

The recommended framework is broken down into the following four stages, as shown in figure 3:

- Requirement Elicitation and Analysis
- Change Identification Analysis
- Changed Requirement Prioritization
- Change Impact Analysis

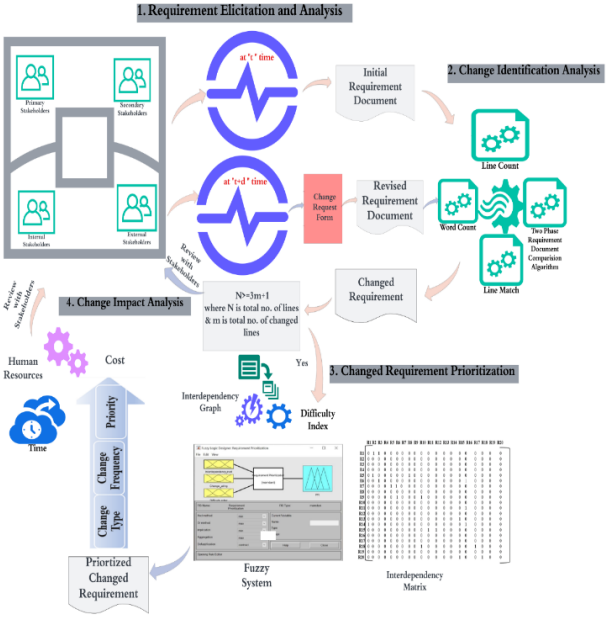


Figure 3: Proposed Framework for RCM

The factors and consequences of requirement change must be understood before the framework can be implemented. The requirement management process might benefit from a deeper comprehension of these factors that have an impact in order to be stronger in the face of disruptions and more adept at navigating toward the expected goal of the changes.

The following subsections each provide a concise explanation of one of the phases of the framework that is suggested in this work:

a). *Requirement Elicitation and Analysis*

Requirement Elicitation and Analysis refers to the elicitation of requirement from the stakeholders. Requirements are collected at time 't' in Initial Requirement Document and after 't+d' time change request comes the Revised Requirement Document is formulated.

b) *Requirement Change Identification*

An accurate identification is required for change prevention and control requirement change. In order to identify the requirement, change in the Software Requirement Document, Two Phase Requirement Document comparison algorithm is proposed.

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 9
***Article Received:** 25 July 2023 **Revised:** 12 September 2023 **Accepted:** 30 September 2023*

Two Phase Requirement Document comparison Tool is developed for finding the count of change lines and identifying changed lines in the Requirement Documents.

There are various attributes for requirement change, major two key attributes of requirement changes are as follows [15]:

- The number of lines that need to be modified
- The number of words that need to be modified

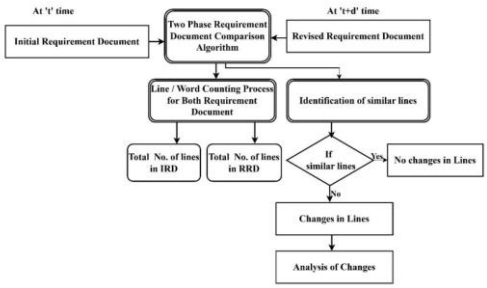


Figure 4: Flow diagram of change identification analysis

Steps for Change Identification

Step 1: Define requirements document 1 at "t" time in terms of individual requirements (requirements Sentences in single lines).

Step 2: Define requirements document 2 at "t+d" time in terms of individual requirements (requirements Sentences in single lines).

Step 3: Calculate the number of lines for each individual requirements of requirements document 1.

Step 4: Calculate the number of lines for each individual requirements of requirements document 2

Step 5: Calculate the match lines of individual requirements line in requirement documents 2

Step 6: Find the match lines and total number of match lines.

Two Phase Requirement Comparison Algorithm

Define $Rd_t \leftarrow \{R1, R2, R3 \ldots Rn\}$; Where Rd_t is a requirements document and Ri is

individual requirement of requirements document (Rd_t) and $Ri \in Rd_t$; $R\,i \subseteq Rd_t$

Define $Rd_{t+d} \leftarrow \{R1, R2, R3 \ldots Rn\}$; Where Rd_{t+d} is a requirements document and Ri is

individual requirement of requirements document (Rd_{t+d}) and $Rij \in Rd_{t+d}$; $Rij \subseteq Rd_{t+d}$

for i ← 1 to n

Li ← lines (Ri) // Li ← line of individual requirement sentence (Ri) in requirement document ($Rd_{t)}$

Lj ← lines (Rj) // Lj ← line of individual requirement sentence (Rij) in requirement document (Rd_{t+d})

Assign L[i] ← {Li1, Li2, Li3, ……., Lin}

Assign L [j]← {Lj1, Lj2, Lj3, ……., Ljn}

// Where L[i] is lines of all requirements sentences in requirements document1 (Rdt)

// Where L[j] is lines of all requirements sentences in requirements document1 (Rd_{t+d})

end for

for i ← 1 to Li

for j ← 1 to Lj

if (Ri[i] != "" || "," || ";")

then NL← R_{ti}[i] // Where NL← New Lines

end for

Initialize count = 0; //Where count is matching count variable

for i ← 1 to n

do

count ← Match (Rdt, Rd_{t+d})

while (count== True)

count ← count +1

Print count

end do while

end for

In the proposed algorithm two attributes are considered. Number of words to be changed and Number of lines to be changed in the Requirement Document. The amendment in the lines is considered as a change in the requirements.

In the first phase lines of both Requirement Document are counted and after finding the count of total number of lines of both Requirement Document, it is assumed that in the idle condition if the numbers of lines are increased in the updated Requirement Document e.g. Revised Requirement Document then requirements are added, and if numbers of lines are

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 9
Article Received: *25 July 2023* ***Revised:*** *12 September 2023* ***Accepted:*** *30 September 2023*

decreased in Revised Requirement Document then it is assumed that requirements are deleted.

c). Changed Requirement Prioritization

It is impossible to always prohibit changes in requirements, since the client can return the software if it does not fully satisfy his needs. Therefore, these changes must be managed extremely well. The only way to determine whether or not a system is effective is to examine whether or not it serves the needs of both its users and its customers [16].

Techniques for requirement prioritizing are utilized to categorize the requirements in priority order. Every stakeholder in the system has concerns that need to be addressed. To begin the process of formulating an approach aimed to the prioritizing of change requirements, the attributes first need to be specified. These change requirements might be prioritized based on a variety of factors [17, 18].

The following measures have been uncovered as having the potential to deliver improved outcomes during the change requirement prioritization process [15]:

i. Interdependency level
ii. Change Rating
iii. Difficulty Index

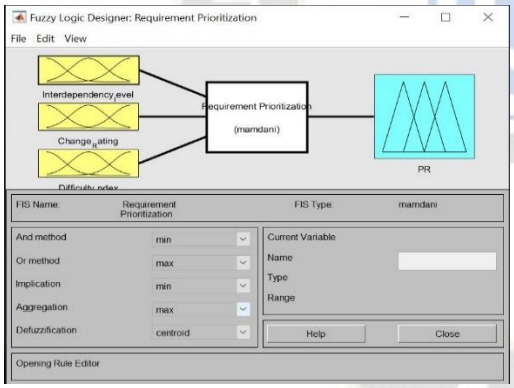


Figure 5: FIS editor for Change Requirement Prioritization

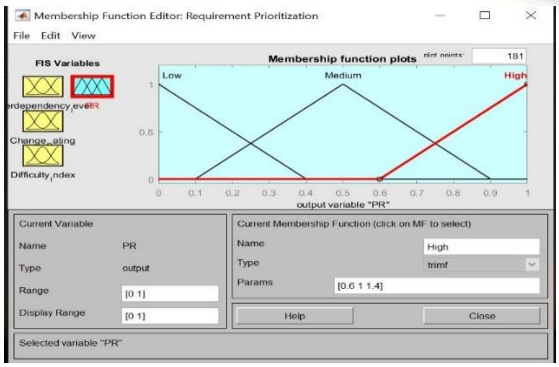


Figure 6: FIS Variables

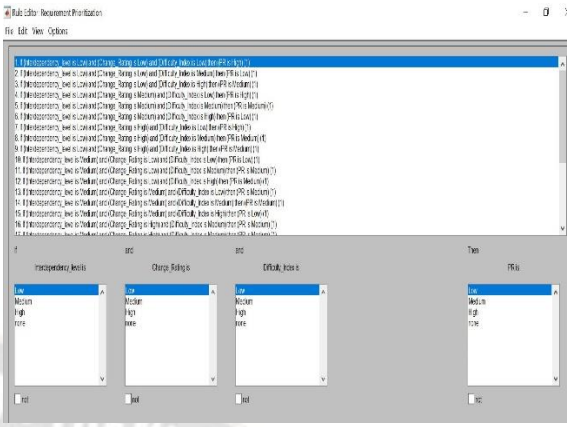


Figure 7: Rule Editor

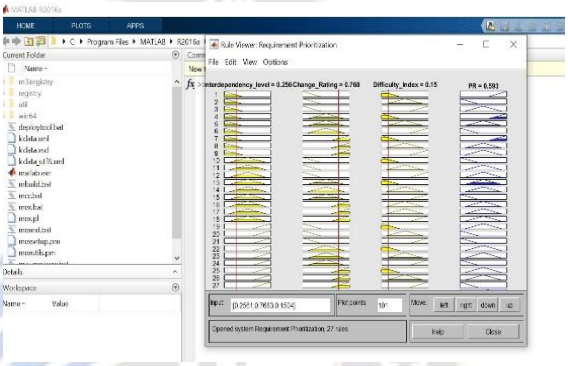


Figure 8: Rule Output

d). Change Impact Analysis

Predictive data may be obtained in a number of ways, and one of them is through change effect analysis. The software maintenance phase is where many modern impact analysis approaches were established [19, 20].

A comprehensive change impact analysis was conducted to assess the potential effects of the identified requirement changes on project constraints such as time, cost, and human resources, taking into consideration the type and priority of each change, units of change, and the frequency of change, with the goal of identifying and mitigating any risks or issues that may arise as a result of the changes.

In terms of adaptability, software is the most flexible part of any system. It evolves not just during the requirements phase but also the rest of the software development lifecycle. To keep customers happy as their needs evolve, it is crucial to effectively manage software updates. When too many changes are approved, the project's completion time increases and total amount must be expended. Customers could get dissatisfied if you choose to ignore the suggested changes. So, it's important for the software project manager to make good

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 9
***Article Received:** 25 July 2023 **Revised:** 12 September 2023 **Accepted:** 30 September 2023*

decisions about how to handle changes while software is being accomplished [21, 22].

Change impact analysis is one of the techniques that may be utilised to give predicted data. For the software maintenance phase, a number of current impact analysis approaches have been established. These techniques presume that all classes in the class artefact are fully built and use the class artefact as an analysis source because it represents the final user requirements. As some classes in the class artefact are still under development or only partially developed, these assumptions are impractical for impact analysis during the software development period [23,24,25].

A survey report says that about 70% of problems happen because the customers' needs change all the time during the process of requirement engineering.70 percent difficulties are attributable bad exercise in requirements management. A research report looked at more than 7,500 software system from more than 300 various business organisations in the United States. It found that only half of the software projects were finished, and almost a 30 percent were never finished. As per a 2018 report by the Standish Group, around 83.9% of software systems are not really completed or fail entirely. As per this report, just 16.1 percent of Software projects were successfully accomplished on schedule and under budget, with the majority of required features. Nearly 52.8% of software projects experienced budget and schedule overruns or were finalized with reduced capability. The rest 31% are deemed failed, which means they were either discontinued or abandoned. Requirement Changes are the primary causes of problematic and unsuccessful initiatives. It has been noted from the above data that changes in requirements have a considerable effect on software development. The software release process, along with the project's timeline, performance, cost, goals, quality, and security, are all directly impacted by requirements changes[26].

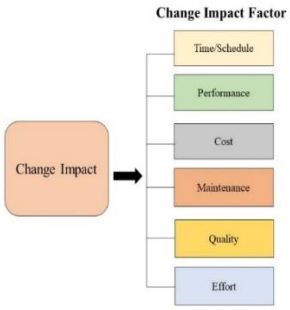


Figure 9: Impact of change on different project parameter

As stated previously, changing requirements has a significant impact on the software development life cycle. Changes to specifications have a significant effect on software release schedules, defect rates, budget, and project performance. The degree to which software project schedules, defect counts, costs, and performance are all negatively correlated with the number of requirements change [27]. The software development life cycle consists of a number of processes, such as the design, development, and testing of the software. Before committing to the development of software, the customer will typically want to know the amount the project will cost, how long it will take, and what all of the internal activities and milestones will be.

The changing of the requirements can have an effect on a variety of different factors [28,29].

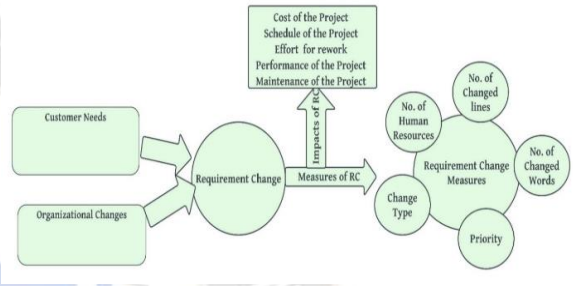


Figure 10: Requirement Change Impact Analysis

REQUIREMENT CHANGE REQUEST FORM

Change Description		
Project Name:		
Change Name:	Addition/Deletion/Modification	
Change Number:		
Requested By:	**Contact:**	**Date:**
Presented To:		
Description of Change:	**Change Frequency**	**Priority rating [Tick One]:**
Requirement 1. **Requirement 2.** **Requirement 3.**		**1. High 2. Medium 3. Low** **2. High 2. Medium 3. Low** **3. High 2. Medium 3. Low**
Reason for Change:		
Impact on Deliverables:		
Impact of Not Approving the Change (and Reason Why):		
Project Manager		
Approved Requirement		
Rejected Requirement		
Date:	**Signature:**	

Figure 11: Requirement Change Request Form

C. *Algorithm for Change Impact Analysis*

// Define change parameters
changeType = ... // type of change (e.g. Add, delete, modify)
changeFrequency = ... // frequency of change (e.g. 1,2,3) maximum 3
changePriority = ... // priority of change (e.g. high, medium, low)
// Identify stakeholders
stakeholders = identifyStakeholders(changeType)
// Perform impact analysis

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 9
***Article Received:** 25 July 2023 **Revised:** 12 September 2023 **Accepted:** 30 September 2023*

impactAnalysis = performImpactAnalysis(changeType, stakeholders)
// Estimate time, human_resource, and cost
human_resourceRequired = estimateHuman_resourceRequired(changeType, priority, change frequency)
timeRequired = estimateTimeRequired(changeType, priority, change frequency)
costOfChange = estimateCostOfChange(changeType ,priority, change frequency)
totalCost = costOfChange + otherProjectCosts
// Create implementation plan
implementationPlan = createPlan(human_resourceRequired, timeRequired, totalCost)
// Execute implementation plan
executePlan(implementationPlan)
// Estimate the human_resource required to implement the change
human_resourceRequired = ...
return human_resourceRequired
function estimateTimeRequired(changeType, priority, change frequency)
// Estimate the time required to implement the change
timeRequired = ...
return timeRequired
function estimateCostOfChange(changeType, priority, change frequency):
// Estimate the cost of implementing the change
costOfChange = ...
return costOfChange
function createPlan(human_resource, time, cost,):
// Create a plan for implementing the change
plan = ...
)
return plan
function executePlan(implementationPlan):
// Execute the implementation plan
execute(implementationPlan)

D. Work Flow of the Framework

The proposed Framework's work flow is shown in the next work flow diagram. It is very helpful for figuring out how the framework works. It is divided into four main blocks that represent the four main steps of the proposed framework. Each block is in charge of doing the task that was given to it. All four parts have something to do with each other.

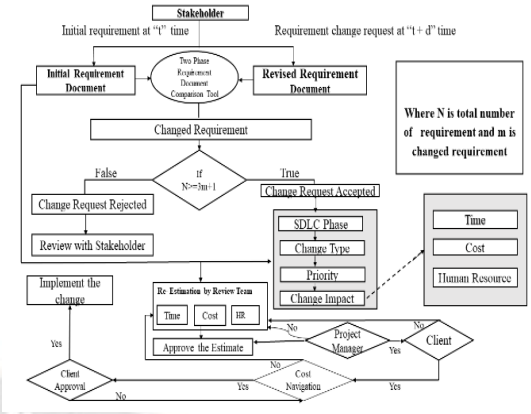


Figure 12: Work Flow of Requirement Change Management Framework

V. IMPLEMENTATION OF PROPOSED FRAMEWORK

In order to determine the likelihood of occurrence for various requirement change variables, software industry professionals were consulted. Intricately designed on Google Form, the survey probed not only the demographic variables including Company Name, Team Size, Total Work Experience and Designation of Experts, as well as the Type of Developed Projects, but also requirement-related parameters, such as the various requirement change factors. Data was acquired from software experts with expertise handling software projects ranging from 2 to 10 years through the posting of an online survey with a carefully prepared questionnaire.

A. *Survey Design*

The questionnaire was compiled by careful examination of the data. On checking the sample
data it was found that the following groups were effectively represented in the survey:

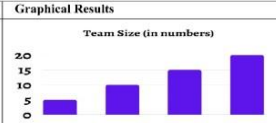
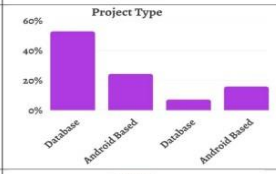
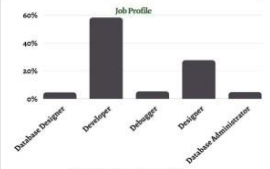
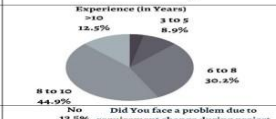
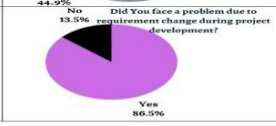
- Team Size of the project
- Project Duration
- Project Type
- Job profile of the experts in the software industry.
- Work Experience of the expert.

In addition to these questions, the experts were requested to express expert view to the requirement-related practices. On the basis of the following questions, the sample data below was formed:

- Responses for problems faced due to requirements.
- Responses for Necessity of early identification of change requirements.

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 9
***Article Received:** 25 July 2023 **Revised:** 12 September 2023 **Accepted:** 30 September 2023*

Table 5: Summary of Survey Result

Survey Questions	Graphical Results	Analysis
Team Size of the project		Less than 15 15.9 16 to 30 29.8 31 to 45 42.2 Above 45 12.1
Project Type		Web based 52.8 Window based 24.3 Database 7.1 Android Based 15.8
Job profile of the experts in the software industry		Database Designer 4.6 Developer 58.3 Debugger 5.3 Designer 27.7 DBA 4.8
Work Experience of the expert		Up to 2 3.5 3 to 5 8.9 6 to 8 30.2 8 to 10 44.9 >10 12.5
Problem faced due to Requirement Change during project development		Yes 86.5% No 13.5%

VI. CASE STUDY

"University Management Information System," an academic MIS project, has been used as a case study to examine the proposed model. The project is briefly stated in Table 7.

Table 6: Summary of UMIS project

Team Size	5 Members
Scheduled Time	8 Months
Project Type	Web Based Project
Platform	WINDOWS
Technology	C# & SQL Server
No. of Functional Requirements	80

The project was developed by a team consisting of 5 members. The requirements were gathered from different institutes which were validated by the concerned head of the institute.

VII. COMPARATIVE ANALYSIS BETWEEN FOUR EXISTING MODELS AND PROPOSED MODEL

Table 7: Four existing models VS Proposed Model

Activities	Dean Leffingwell and Widrig model	Niazi et al. model	Hussain & Clear model	CRCM model	Proposed Model
	2000	2008	2012	2017	2023
1. Plan for change	√		√	√	√
2. Baseline requirements	√		√	√	√
3. Use a change control system to capture changes	√	√	√	√	√
4. Change impact on cost	√			√	√
5. Change Frequency					√
6. Change Priority					√
7. Documenting the actions and observations				√	√
8. Determine the type of change		√			√
9. Change impact on Human Resources					√
10. Change impact on time					√

VIII. VALIDATION AND VERIFICATION OF THE PROPOSED FRAMEWORK

A. *Introduction to Finite State Machine*

A computational model for the static and dynamic behaviour of a software system is presented by the finite state machine (FSM). It is a conceptual machine that builds a finite number of states, and it generates one state at a time by reading input symbols. The number of states that can be built by this machine is limited. The FSM begins its operation at the initial state and continues its work until it reaches the final state [33]. It is capable enough to take any input string that has a limited number of alphabets in it. An input string may be accepted by the finite state automaton or it may be rejected [34].

B. Fundamental of FSM

The mathematical model known as the finite state machine is typically utilised during the process of developing computer programmes. In addition to this, it is recognised as a deterministic finite automaton. One way to define a finite state machine

is as follows:

M = (S, Σ, δ, s0, F) where

- S means a finite set of state, Σ means a finite set of input symbols.
- A transition function denoted by δ takes an input symbol and a state as arguments and returns a state. In the graphical representation, δ is represented by an arc linking two states. If s be a state and x be an input symbol, then δ (s, x) defines state l that has an arc labeled x from s to l.
- s_0 means an initial state that a state of S.
- F means final state.

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 9
***Article Received:** 25 July 2023 **Revised:** 12 September 2023 **Accepted:** 30 September 2023*

From this description of automata, a finite state machine is created in which states match up to the variable S and all the transitions and input symbols corresponds to the variable δ and Σ, respectively [30,31].

Requirement Change is a frequent phenomenon in industrial software development. It is deemed to be a main cause of risk to the management of large and complex software projects. Effective Requirement management techniques can be helpful in controlling the changing software requirements and lower the costs due to these changing requirements [32]. In this thesis an effective framework for managing requirement change during SDLC is proposed that deals with these precarious changing requirements. Finally, the validation has been performed through finite state machine. For the purpose, the method to validate the framework using FSM is illustrated with the conception of transition table. FSM is used as to develop the technique to substantiate the correctness of the proposed framework. This method generates test cases through FSM that is very effective and consistent method which doesn't sustain invalid test cases [32].

1). Design State transition diagram of the proposed framework

Software Requirement is assumed to be the initial as well as the final state of the framework; this state is equivalent to "s0".

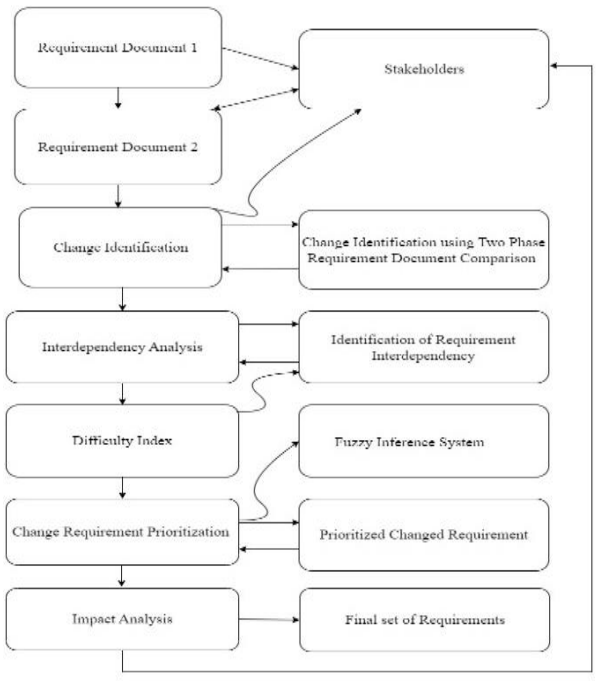


Figure 13: State Transition diagram for proposed framework.

2) Design the Finite State Machine (FSM)

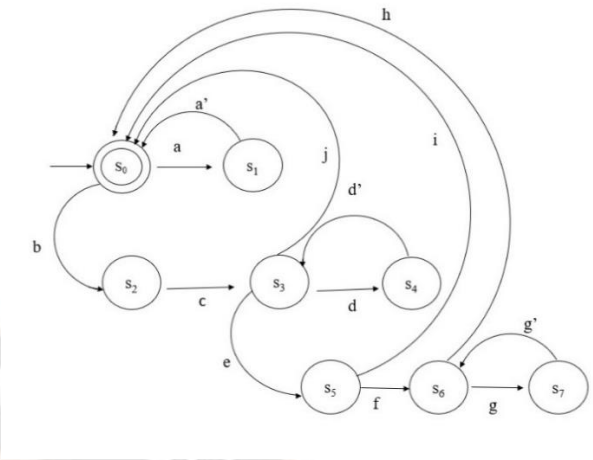


Figure 14: State Transition Diagram

3) Transition Table for FSM

From the above FSM, {a, b, b′, b″, c, ……j} events describe the transformation of states from one state to another. These events are deliberated as terminals for this finite state machine and the set of states {s0, s1, s2, s3, s4, s5} are implied as nonterminal. The state q0 is the initial and the final state. Following diverse productions have been induced for the finite state machine and the consequent transition table is shown below in table.

Table 8: State Transition Table

Current State	Event	Next State
s0	a	s_1
s0	b	s_2
s_2	c	s_3
s_3	d	s_4
s_3	e	s_5
s_3	j	s0
s_4	d'	s_3
s_5	f	s_6
s_5	i	s0
s_6	g	s_7
s_6	h	s0
s_7	g'	s_6

4)Production Rules

Several production rules are generated for the above finite state machine as mentioned below:

- $s_0 \rightarrow a\ s_1 \mid b\ s_2 \mid i\ s_0 \mid h\ s_0$
- $s_1 \rightarrow a'\ s_2$
- $s_2 \rightarrow c\ s_3$
- $s_3 \rightarrow d\ s4 \mid e\ s5 \mid j\ s_0$
- $s_4 \rightarrow d'\ s_3$
- $s_5 \rightarrow f\ s_6 \mid i\ s_0$
- $s_6 \rightarrow g\ s_7 \mid h\ s_0$
- $s_7 \rightarrow g'\ s_6$

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 9
***Article Received:** 25 July 2023 **Revised:** 12 September 2023 **Accepted:** 30 September 2023*

These productions rules can be defined as followings:

- $s_0 \rightarrow$ a s1: The FSM will go from state s0 to state s1 if it gets the input symbol 'a' while it is in the state s0.
- $s_1 \rightarrow$ a' s_2: If the FSM is in state s1 and it gets the input symbol 'a', then it transitions to state s2; otherwise, it stays in state s1.
- $s_0 \rightarrow$ b s_2: The FSM will go from state s0 to state s2 if it gets the input symbol 'b' while it is in the state s0.
- $s_2 \rightarrow$ c s_3The FSM moves from state s2 to state s3 when it gets the input symbol 'c' if it is already in the state s2.
- $s_3 \rightarrow$ d s_4: The FSM will go from state s3 to state s4 if it gets the input symbol 'd' while it is in the state s3.
- $s_3 \rightarrow$ e s_5: The FSM will go from state s3 to state s5 if it gets the input symbol 'e' when it is in the state s3.
- $s_3 \rightarrow$ j s_0: If the FSM is in state s3 and receives input symbol 'j', it transitions to state s_0.
- $s_4 \rightarrow$ d' s3The FSM will go from state s4 to state s3 if it gets the input symbol 'd' while it is in state s4; otherwise, it will remain in state s4.
- $s_5 \rightarrow$ f s6: If the FSM is now in state s5 and it gets the input symbol 'f,' then it will go on to state s6.
- $s_5 \rightarrow$ i s0: The FSM will go from state s5 to state s0 when it gets the input symbol I if it is currently in state s5.
- $s_6 \rightarrow$ g s_7: When the FSM is in state s6 and it gets the input symbol 'g,' it will go to state s7 if the situation is met.
- s6 $\rightarrow$ h s0: The FSM will go from state s6 to state s0 when it gets the input symbol 'h' if it is already in state s6.
- $s_7 \rightarrow$ g' s6: The FSM will go from state s7 to state s6 if it gets the input symbol 'g' while it is in the state s7.

where s_0, s_1, s_2, s_3, s_4, s_5, s_6, and s_7 are the state symbols and 'a', 'a'', 'b', 'c', 'd', 'd'', 'e', 'f', 'g', 'g'', 'h', and 'i' are the input symbols.

10. Conclusion and future work

The purpose of this study was to offer a thorough knowledge of requirement change (RC) and its influence on software development initiatives. The introduction emphasised the importance of requirements in the development process and the need of managing requirement changes effectively. The causes of RC were investigated in order to offer insight on both internal and external variables that lead to changes in project needs. Furthermore, change effect analysis was addressed as an important technique for assessing the repercussions of requirement modifications.

The study's findings highlight the necessity of proactive requirement management in minimising the harmful consequences of RC. Understanding the causes of RC allows project managers and stakeholders to take preventative efforts to reduce the risks associated with changes. Furthermore, using change effect analysis methodologies allows project teams to examine the possible consequences of requirement modifications, allowing them to make educated judgments about whether to accept or reject a suggested change.

This study emphasizes on small project which can be enhanced with large project in future.

ACKNOWLEDGMENT

This work is acknowledged under Integral University manuscript No IU/R&D/2023 - MCN0001968.

REFERENCES

[1] Abd, M., & Latif, E. L. (2016). *Identify and Manage the Software Requirements Volatility*. *7*(5), 64–71.

[2] Dev, H., & Awasthi, R. (2012). A Systematic Study of Requirement Volatility during Software Development Process. *International Journal of Computer Science Issues*, *9*(2), 527–533.

[3] Passenheim, O. (2010). Change Management Change Management. *Introduction to Information Security*, *1*(3), 1–50. bookon.com

[4] Errida, A., & Lotfi, B. (2021). The determinants of organizational change management success: Literature review and case study. *International Journal of Engineering Business Management*, *13*, 1–15. https://doi.org/10.1177/18479790211016273

[5] Rodrigues da Silva, A., & Olsina, L. (2022). Special Issue on Requirements Engineering, Practice and Research. *Applied Sciences (Switzerland)*, *12*(23). https://doi.org/10.3390/app122312197

[6] Ahmad, S. (n.d.). *A Systematic Literature Review on Requirement Change Management Challenges Faced By Developers*.

[7] Sadia, H. (2014). *Requirement Risk Identification : A Practitioner ' s Approach*. *102*(15), 13–15.

[8] Afaq, S. A. (2020). *Influences of Requirement Change on Software Failure*. *4178*, 4178–4184.

[9] Bano, M., Imtiaz, S., Ikram, N., Niazi, M., & Usman, M. (2012). Causes of requirement change - a systematic

literature review. *16th International Conference on Evaluation & Assessment in Software Engineering (EASE 2012)*, 22–31. https://doi.org/10.1049/ic.2012.0003

[10] Melo, A., Fagundes, R., Lenarduzzi, V., & Santos, W. B. (2022). Identification and measurement of Requirements Technical Debt in software development: A systematic literature review. *Journal of Systems and Software*, 111483.

[11] Ahmad, J., Ghazal, T. M., Khan, A. W., Khan, M. A., Inairat, M., Sahawneh, N., & Khan, F. (2022). Quality requirement change management's challenges: An exploratory study using slr. *IEEE Access*, *10*, 127575–127588.

[12] Dasanayake, S., Aaramaa, S., Markkula, J., & Oivo, M. (2019). Impact of requirements volatility on software architecture: How do software teams keep up with ever-changing requirements? *Journal of Software: Evolution and Process*, *31*(6), 1–21. https://doi.org/10.1002/smr.2160

[13] Madampe, K., Hoda, R., & Grundy, J. (2023). A Framework for Emotion-oriented Requirements Change Handling in Agile Software Engineering. *IEEE Transactions on Software Engineering*.

[14] Park, S., Maurer, F., Eberlein, A., & Fung, T.-S. (2010). Requirements attributes to predict requirements related defects. *Proceedings of the 2010 Conference of the Center for Advanced Studies on Collaborative Research*, 42–56.

[15] Afaq, S. A., & Faisal, M. (2021). *An Efficient Approach For Software Requirement Change Identification*. *18*(3), 1919–1926.

[16] Dahiya, O., & Solanki, K. (2021). An efficient requirement-based test case prioritization technique using optimized TFC-SVM approach. *International Journal of Engineering Trends and Technology*, *69*(1), 5–16. https://doi.org/10.14445/22315381/IJETT-V69I1P202

[17] Lachmann, R. (2020). *12.4 - Machine Learning-Driven Test Case Prioritization Approaches for Black-Box Software Testing*. 300–309. https://doi.org/10.5162/ettc2018/12.4

[18] Ruby, & Balkishan. (2015). Fuzzy Logic based Requirement Prioritization (FLRP) - An Approach. *International Journal of Computer Science and Technology*, *6*(3), 61–65.

[19] Sharma, P., & Singh, J. (2018). Systematic literature review on software effort estimation using machine learning approaches. *Proceedings - 2017 International Conference on Next Generation Computing and Information Systems, ICNGCIS 2017*, 54–57. https://doi.org/10.1109/ICNGCIS.2017.33

[20] Wang, J., Li, J., Wang, Q., Zhang, H., & Wang, H. (2012). A simulation approach for impact analysis of requirement volatility considering dependency change. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *7195 LNCS*, 59–76. https://doi.org/10.1007/978-3-642-28714-5_6

[21] Nurmuliani, N., Zowghi, D., & Williams, S. P. (2006). Requirements Volatility and Its Impact on Change Effort: Evidence Based Research in Software Development Projects." Proc. Australian Workshop on Requirements Engineering (AWRE 2006), Adelaide, Australi. *Australian Workshop on Requirements Engineering*. http://www.researchgate.net/publication/228946043_Requirements_volatility_and_its_impact_on_change_effort_Evidence-based_research_in_software_development_projects/file/9c960520ecb3089ce7.pdf

[22] Tung, K. T., Hung, N. D., Thi, L., & Hanh, M. (2015). *A Comparison of Algorithms used to measure the Similarity between two documents*. *April*.

[23] Kama, N., & Azli, F. (2012). A change impact analysis approach for the software development phase. *Proceedings - Asia-Pacific Software Engineering Conference, APSEC*, *1*, 583–592. https://doi.org/10.1109/APSEC.2012.89

[24] Haleem, M., & Beg, M. R. (2015). Impact analysis of requirement metrics in software development environment. *2015 IEEE International Conference on Electrical, Computer and Communication Technologies (ICECCT)*, 1–6.

[25] Sharif, B., Khan, S. a, & Bhatti, M. W. (2012). Measuring the Impact of Changing Requirements on Software Project Cost : An Empirical Investigation. *International Journal of Computer Science*, *9*(3), 170–174.

[26] Haleem, M., & Farooqui, F. (2021b). International Journal of Cognitive Computing in Engineering Tackling Requirements Uncertainty in Software Projects : A Cognitive Approach. *International Journal of Cognitive Computing in Engineering*, *2*(October), 180–190. https://doi.org/10.1016/j.ijcce.2021.10.003

[27] Goknil, A., Kurtev, I., & Berg, K. van den. (2016). *A Rule-Based Change Impact Analysis Approach in Software Architecture for Requirements Changes*. http://arxiv.org/abs/1608.02757

[28] Ahmad, Z., Hussain, M., Rehman, A., Qamar, U., & Afzal, M. (2015). *Impact Minimization of Requirements Change in Software Project through*

Requirements Classification. 4–8.

[29] Haleem, M., & Farooqui, F. (2021a). International Journal of Cognitive Computing in Engineering Cognitive impact validation of requirement uncertainty in software project development. *International Journal of Cognitive Computing in Engineering*, *2*(October 2020), 1–11. https://doi.org/10.1016/j.ijcce.2020.12.002

[30] Vayadande, K. B., Sheth, P., Shelke, A., Patil, V., Shevate, S., & Sawakare, C. (2022). Simulation and Testing of Deterministic Finite Automata Machine. *International Journal of Computer Sciences and Engineering*, *10*(1), 13–17.

[31] Yankovskaya, A., & Yevtushenko, N. (1997). Finite state machine (fsm)–based knowledge representation in a computer tutoring system. *New Media and Telematic Technologies for Education in Eastern European Countries*, 67–74.

[32] Muqeem, M., & Beg, M. R. (2014). Validation of requirement elicitation framework using finite state machine. *2014 International Conference on Control, Instrumentation, Communication and Computational Technologies (ICCICCT)*, 1210–1216.

[33] Lee, D., & Yannakakis, M. (1996). Principles and methods of testing finite state machines-a survey. *Proceedings of the IEEE*, *84*(8), 1090–1123.

[34] Grieskamp, W., Gurevich, Y., Schulte, W., & Veanes, M. (2002). Generating finite state machines from abstract state machines. *Proceedings of the 2002 ACM SIGSOFT International Symposium on Software Testing and Analysis*, 112–122.