International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 5
DOI: https://doi.org/10.17762/ijritcc.v11i5.6589
***Article Received**: 20 February 2023 **Revised**: 27 March 2023 **Accepted**: 10 April 2023*

Enhancing Requirements Change Request Categorization and Prioritization in Agile Software Development Using Analytic Hierarchy Process (AHP)

Kashif Asad[1], Dr. Mohd. Muqeem[2]
[1]Integral University: Department of Computer Application
Lucknow, India
kashif.asad007@gmail.com
[2]Integral University: Department of Computer Application
Lucknow, India
muqeem@iul.ac.in

Abstract— Software development now relies heavily on agile methods, which call for the efficient administration and prioritization of change requests. In order to improve requirement prioritization using the Analytic Hierarchy Process (AHP) in Agile methods, this study article presents a new framework for classifying software requirements into Small Change Requests (SCRs) and Large Change Requests (LCRs). The paper examines the difficulties associated with requirement prioritization and categorization in Agile settings and offers a methodical system for dividing change requests into categories based on complexity, impact, and timeline. In order to provide a thorough grasp of the project scope and objectives, the framework considers both functional and non-functional needs. A case study containing several Agile software development projects is used to evaluate the performance of the suggested categorization and prioritization model. According to the findings, the combination of SCR and LCR categorization with AHP enables more effective teamwork and greater matching of development goals with partner objectives. The research also shows that the suggested framework's integration into the Agile development process results in a more efficient decision-making process, less time wasted on talks, and improved resource distribution. The model aids in risk mitigation by allowing a methodical and quantifiable approach to requirement prioritization. These risks are related to quick changes in project scope and changing client requirements. By presenting a fresh framework for requirement categorization and prioritization, this study adds to the current discussion on successful requirement management in Agile methods. Agile software development projects become more effective and adaptable overall thanks to the incorporation of AHP, which guarantees a more methodical and objective prioritization process. This study has the potential to greatly improve the administration of shifting needs and user expectations in Agile settings by offering a structured method to classify and rank change requests.

Keywords- Agile Software Development, Requirement Change, Requirement Categorization, Impact Analysis, Requirement Prioritization, Analytical Hiearchy Process, Change Request.

I. INTRODUCTION

Agile software development is common for handling tasks with quickly changing requirements and goals. Agile methods help software development teams react to changing needs and new knowledge. Changes in requirements are common in agile software development and can be brought on by a variety of factors, including corporate, market, client, or software worker expertise growth [13]. Agile practitioners face challenges classifying and prioritizing requirements, especially when working with varied change requests of different ranges and effects. In contrast, agile software development gradually elaborates the product pursuing user happiness and "welcomes changing requirements, even late in development."[12].

This study paper presents a new framework for classifying agile software development change requests into small change requests (SCRs) and large change requests (LCRs) and using the Analytic Hierarchy Process (AHP) to rank these requirements. The framework improves decision-making and resource and time allotment, improving project results and software quality. This paper introduces agile software development methods and the obstacles teams face when handling and prioritizing change requests. Then, the criteria for separating SCRs from LCRs are examined, considering factors like the intricacy of the change, its impact on the current system, and the time and effort needed to execute it. The process of selecting which requirements need to be implemented ahead of the others requires that a prioritization of the requirements be performed [4]. We'll discuss this categorization's reasoning and project management consequences. In rapid software development, the Analytic Hierarchy Process is a reliable and efficient demand ranking method. The AHP's methodical strategy to paired evaluations and weighted decision-making and its fit for agile projects' dynamic problems will be explored. The study will then apply the AHP method to the classified SCRs and LCRs to rate change requests by project relevance and impact. A case study will demonstrate how agile teams can use the AHP-based framework to rank change requests. The case study will show how the proposed framework improves efficiency, resource sharing, and uncertainty-based decision-making. This study will also explore potential drawbacks and alternatives to the proposed AHP-based framework. The talk will also explore the framework's flexibility to handle projects of various sizes and intricacies and its adaptability to agile methods and corporate settings. In conclusion, this paper proposes a novel AHP-based framework for classifying and prioritizing change requests in agile software development. This strategy helps agile teams handle demand management and react to changing needs while delivering high-quality software products on time. This paper's insights and suggestions can improve agile software development by giving practitioners tools for handling change and ensuring project success.

II. RELATED WORK

R. Thakurta [5] provided a scenario-based quantified framework for prioritizing nonfunctional requirements. This approach failed to integrate new or changed requirements, and the assessment had validation issues. The "Requirements Prioritizer" multi-criteria decision-making device [6] prioritized requirements from any place. The authors presented their support to the process of prioritizing stakeholders by ordering requirements according to the importance of the characteristics that were supplied by the pertinent stakeholders. Every one of the requirements has to be able to stand on its own. This strategy, which has been proposed, addresses both the order inversion and the interdependence problems [7]. Gershenson and Stauffer [8] created a framework for the categorization of the different requirements that corporations must meet. The requirements of the corporation are derived from various internal sources, such as marketing, finance, manufacturing, and service, and they are intended to represent the requirements of the corporation with regard to product development. The impacts of using agile techniques in project management, as well as the individuals participating in the project and their applicability, were analyzed by Michael Coram and his colleagues. In order to adapt to the adjustments, the Agile Methods suggested taking a pragmatic strategy. When utilized in the appropriate context, agile techniques can be of great assistance [10]. For the problems that arise during agile software development, Veerapaneni Esther Jyothi et al. suggested a joint and original approach [15].

In order to determine the characteristics of requirement changes, Ghosh [14] carried out an experimental study on 30 software development initiatives that involved requirement changes. The author employed a regression-based forecast algorithm to determine an EV value at any time. In Scrum, there is neither a specified method nor a structure that is established specifically for the purpose of forecasting requirement fluctuation. Baxter et al. [11] developed a framework for the integration of design knowledge reuse and requirements management. This framework enables the application of requirements management as a dynamic process. Hussain et al. [9] provided an overview of the process of change management. The process of change management begins with the submission of change proposals; the change proposition is made accessible to all stakeholders. The implementation of software that is capable of managing the change is the second stage. The third step is to analyze how the suggested change will affect the system. During the process of change management, one of the most essential activities is to determine the effects that the requested change will have on the organization and to estimate the amount of revenue that will be required to carry out the requested change. The consolidated structure of the change management process was established after the three previously mentioned stages had been identified. Quesf, A. [17] addresses the requirements traceability challenge that arises in agile software development, as well as the relationships that exist between the refactoring and traceability processes, as well as the effect that these processes have on one another. Mueller Investigate the effect of requirements changes on development output in an agile-scrum software development process to determine if there is a link between development effort and requirement changes [26].

A software change classification has been proposed by Buckley et al. [31], and it is founded on characterizing the mechanisms of change as well as the variables that influence software change. According to Khan et al. [33], the communication activity is an essential part of the RCM process. Furthermore, during the implementation of the suggested requirement changes in the agile GSD paradigm, the significance of this activity grows significantly. Akbar et al. [2] noted that requirements can alter throughout software development, from requirements elicitation to release. Nurmuliani et al. [3] describe it "requirement change" because it means changing to meet the needs of customers, partners, companies, and the workplace as they change. Forrester [28] states that agile requirements management tools need to have seven features in order to be effective. These features include the ability to support agile methodologies, traceability and change impact, user story creation and management, social collaboration,

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 5
DOI: https://doi.org/10.17762/ijritcc.v11i5.6589
***Article Received**: 20 February 2023 **Revised**: 27 March 2023 **Accepted**: 10 April 2023*

visualization of requirements, scalability, and integrations with other tools.

McGee and Greer [30] came up with a taxonomy that is structured on the basis of the origin of RC and their categorization in accordance with the change source domain. The classification makes it possible for software practitioners to differentiate between the variables that contribute to requirements uncertainty, which ultimately results in improved visibility of change identification. By combining the agile methodology with the CMMI, Glazer et al. [32] provided Web software development companies with a means to create high-quality systems without sacrificing their adaptability to change. Web application creation is the foundation of I4.0 technologies. A comprehensive literature analysis on requirements engineering practices and difficulties in the setting of ASD was performed by Inayat et al. [29]. The study revealed that there is a total of seventeen different RE practices in agile, as well as five problems with conventional techniques that were solved by agile requirements engineering, and eight problems with RE in agile itself.

III. REQUIREMENTS CHANGE TAXANOMY

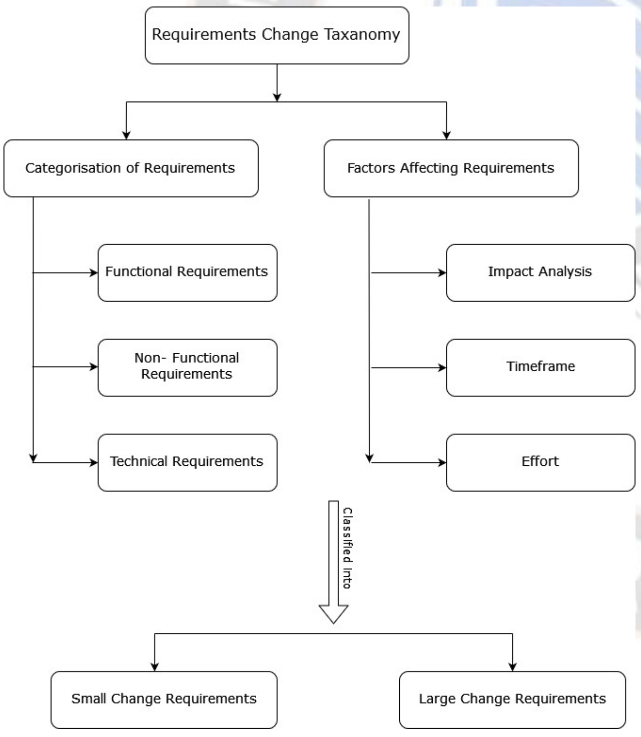
Requirements Change Taxonomy
Categorisation of Requirements
Factors Affecting Requirements
Functional Requirements
Non- Functional Requirements
Technical Requirements
Impact Analysis
Timeframe
Effort
Classified into
Small Change Requirements
Large Change Requirements

Figure 1. Requirements Change Taxonomy

A. Categorization Of Change Requirements

In Agile, changes are expected and handled with the help of iteration meetings, iterative development, and an ordered product list. The agile approach is a useful tool for managing requirements in an incremental method, which becomes increasingly important as the nature of the requirements changes and becomes more unpredictable [34]. In order to effectively solicit, record, implement, and handle changes to Requirements, a method known as Change Requirement Management (CRM) is employed. Analysis, assessment, execution, and change request management are the core tenets of customer relationship management [27]. Any proposed alterations are thoroughly analyzed before being integrated into the project plan. Agile approaches value adaptability and rapid response. The main reasons for Requirement Change Management are changing customer needs, changing market trends, and changing company or business needs [20]. The RM system should be used to document any and all activities in a corporation that are connected to a requirement [23]. There are various categories of requirements, such as functional, non-functional, and quality aspects of a product. Frequently, there may be dependencies between various requirements [35]. Requirements can be classified into following-:

i) Functional Requirements are descriptions of what a software system should do and how it should respond to particular inputs or actions. These criteria describe the functions, services, and features that the software program must provide to satisfy the company requirements and customer demands. User interfaces, data processing and storing, user administration, monitoring and statistics, and system interaction are all examples of functional needs. The functional requirements provide a definition of the behaviors or activities that the system ought to be capable of supporting [36].

ii) Non-Functional Requirements: Non-functional requirements describe the characteristics or traits of the software system, such as its usefulness, speed, dependability, security, scaling, and maintainability. The effectiveness of the system's performance of its duties is more important than its usefulness. Non-functional criteria can also specify limits on the system, such as legal conformance or interoperability with other systems. The non-functional requirements refer to the characteristics of the system that are not immediately related to the functions that are performed by the system [37]. Non-functional criteria can include things like reaction speed, access, data protection and security, and simplicity of upkeep.

iii) Technical Requirements: Technical requirements are descriptions of the hardware, software, and other technological components needed to create, implement, and manage the software system. In this context, "requirements" refer to the technologies, tools, and platforms that will be used to construct the system, as well as the hardware and infrastructure that will be required to support it. Performance, security, and scaling criteria may also be specified in technical specifications. The technology, database management system, operating system,

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 5
DOI: https://doi.org/10.17762/ijritcc.v11i5.6589
***Article Received**: 20 February 2023 **Revised**: 27 March 2023 **Accepted**: 10 April 2023*

server hardware, and network architecture are a few examples of technological needs.

TABLE I. DESCRIPTION OF REQUIREMENT TYPES

Requirement Type	Requirement Description
Functional Requirements	Allow customers to search for products by keyword
Functional Requirements	Allow customers to add products to a shopping cart
Functional Requirements	Allow customers to create an account and save their info
Functional Requirements	Allow customers to view their order history
Non-Functional Requirements	The website should load within 3 seconds
Non-Functional Requirements	The website should be accessible to people with disabilities
Non-Functional Requirements	The website should be secure and protect customer data
Technical Requirements	The website should be built using ReactJS
Technical Requirements	The website should be hosted on AWS
Technical Requirements	The website should use a PostgreSQL database

B. Factors Affecting Categorization Of Change Requirements

i) Impact analysis: Suppose a client asks a change to a software product's user interface. To determine how the change will affect the software's current operation, the development team must first perform an impact study. They might find that the change will necessitate substantial alterations to the underlying code, adding to the effort and possibly postponing the delivery of the product.
ii) Timeframe: Let's say a software product proprietor asks for a new function to be introduced. The development team must assess the timeframe needed to execute the change and any possible impact it might have on the project's general schedule. If the change is complicated and will take a long time to execute, the team may recommend breaking it down into smaller, more doable jobs to ensure that it is finished within the project's timeframe.
iii) Effort: The amount of work, time, and resources required to execute a change proposal in a project are referred to as effort when discussing the variables influencing the classification of change needs. The task completed by one person in a given number of hours or days is usually defined in person-hours or person-days.

C. How Change Requirements in Agile categorize into Small Change Request and Large Change Request

We can use the following parameters to classify change proposals into Small and Large based on the impact analysis on the project and time frame: The change request will be categorized as a Small Change Request if it has a minimal impact on the project and can be executed quickly. The details regarding the change request, along with all of the specifics of the change request, are included in the document for the change request [16]. The change request will be categorized as a Large Change Request if it has a significant impact on the project and takes more time to execute.

Impact Analysis is the process of determining the prospective impact of a proposed change to a software system. Impact Analysis considers a variety of variables, including the impact on functionality, performance, scalability, security, and user experience, among others. These impact ratings are typically given based on a thorough analysis of the proposed change and its potential impact on various aspects of the software system.

- Functionality: This impact rating could be based on how much the proposed change would affect the overall functionality of the software system. A change that introduces new functionality or significantly alters existing functionality could be given a higher rating than a change that only makes minor adjustments.
- Performance: This impact rating could be based on how the proposed change would affect the speed or responsiveness of the software system. A change that would significantly slow down the system or make it less responsive could be given a higher rating than a change that has little impact on performance.
- Scalability: This impact rating could be based on how the proposed change would affect the ability of the software system to handle larger volumes of data or users. A change that would significantly limit the scalability of the system could be given a higher rating than a change that has little impact on scalability.
- Security: This impact rating could be based on how the proposed change would affect the overall security of the software system. A change that introduces new security risks or significantly weakens existing security measures could be given a higher rating than a change that has little impact on security.
- User Experience: This impact rating could be based on how the proposed change would affect the overall user experience of the software system. A change that significantly improves or degrades the user experience could be given a higher rating than a change that has little impact on user experience.

These ratings are typically assigned by a team of experts who have a deep understanding of the software system and the proposed change. The ratings are based on a combination of technical analysis, experience, and judgment. Once the impact ratings are assigned, they can be used to calculate the overall impact analysis score using the formula mentioned earlier. The formula to calculate Impact Analysis Score can be expressed as:

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 5
DOI: https://doi.org/10.17762/ijritcc.v11i5.6589
***Article Received**: 20 February 2023 **Revised**: 27 March 2023 **Accepted**: 10 April 2023*

$$\text{Impact Analysis Score} = \sum (Wi * Xi)$$

Where Wi is the Weightage Factor for Each Impact Area (functionality, performance, scalability, security, user experience)
Xi = The impact rating for each impact region on a range of 1 to 10. (where 1 is the lowest impact and 10 is the highest impact)
To demonstrate how Impact Analysis Score is determined, let's look at an example. Consider the following when examining a modification proposal that has the following impact on a software application: Let's suppose that we have given each impact region the following weights:
Functionality: 0.3, Performance: 0.2, Scalability: 0.1, Security: 0.3, User Experience: 0.1

IV. RESEARCH METHOD

This section provides an overview of the research technique that was used in the study, which was the application of the Analytic Hierarchy Process (AHP) for the purpose of classifying change requests in Agile software development projects and assigning priorities to those change requests.

A. DATA COLLECTION

In a digital world that is becoming more competitive, an e-commerce site needs to be able to change and get better all the time to keep and gain clients. This case study looks at how to find, prioritize, and apply different change requests (CRs) to make an e-commerce website better.

TABLE II. DESCRIPTION OF REQUIREMENT TYPES

Change Request	Description of Change Request
CR-001	Add new payment method
CR-002	Fix search functionality
CR-003	Implement customer reviews for products
CR-004	Optimize website performance
CR-005	Update user interface design
CR-006	Improve inventory management system
CR-007	Fix broken links

Each impact region is given a weightage factor, which is a figure that represents its relative significance or importance in the total impact analysis. The total of all weightage variables should equal 1 (or 100%), and it is typically expressed as a percentage or numeric value. We'll also assume the following impact ratings for each change request.

TABLE III. IMPACT RATING OF CHANGE REQUEST

Change Request	Functionality	Performance	Scalability	Security	User Experience
CR-001	9	6	3	8	5
CR-002	5	9	4	7	8
CR-003	8	5	6	9	7
CR-004	7	8	5	6	9
CR-005	4	3	2	5	8
CR-006	8	7	9	8	4
CR-007	2	2	1	2	3

Using the formula, we can calculate the Impact Analysis Score for each change request as follows:

TABLE IV. RESULTANT IMPACT ANALYSIS SCORE

Change Request	Impact Analysis Score
CR-001	7.3
CR-002	6.8
CR-003	7.4
CR-004	7.2
CR-005	4.2
CR-006	7.5
CR-007	1.9

B. Implementing Impact Analysis Score And Time Frame For Change Request Categorization In ASD

The following parameters can be used to illustrate this classification:

- Small Change Request: Impact Analysis Score ≤ 6, Timeframe ≤ 2 sprints, and Effort Score ≤ 4
- Large Change Request: Impact Analysis Score > 6, Timeframe > 2 sprints, or Effort Score > 4

Estimate the Effort in person-hours for each change request and calculate the Effort Score (E) using the formula:
E = Person-hours required / 8.

TABLE V. EFFORT SCORE

Change Request	Person-hours	Effort Score
CR-001	24	6.0
CR-002	56	2.0
CR-003	40	5.0
CR-004	16	8.0
CR-005	24	3.0
CR-006	80	10.0
CR-007	8	1.0

We have the following Timeframe in sprints for each change request.

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 5
DOI: https://doi.org/10.17762/ijritcc.v11i5.6589
***Article Received**: 20 February 2023 **Revised**: 27 March 2023 **Accepted**: 10 April 2023*

TABLE VI. TIMEFRAME OF CHANGE REQUEST

Change Request	Timeframe (sprints)
CR-001	3
CR-002	1
CR-003	2
CR-004	4
CR-005	1
CR-006	5
CR-007	1

Now let's categorize the change requests based on the criteria defined above:

TABLE VII. RESULTANT CHANGE REQUEST CATEGORIZATION

Change Request	Impact Analysis Score	Timeframe (sprints)	Effort Score	Category
CR-001	7.3	3	6.0	Large Change Request
CR-002	6.8	1	2.0	Large Change Request
CR-003	7.4	2	5.0	Large Change Request
CR-004	7.2	4	8.0	Large Change Request
CR-005	4.2	1	3.0	Small Change Request
CR-006	7.5	5	10.0	Large Change Request
CR-007	1.9	1	1.0	Small Change Request

The development team is now able to successfully prioritize and handle the change requests based on the Impact Analysis Score, Timeframe, and Effort Score.
Change Requests CR-001, CR-002, CR-003, CR-004, and CR-006 will go through the Requirement Prioritization process, whereas Change Requests CR-004 and CR-007 will go straight to the Implementation process.

C. Causes for Variation in Agile Methodology Requirements

TABLE VIII. FREQUENCY OF REQUIREMENT CHANGES

Frequency	Percentage of Projects
Rarely	10%
Occasionally	40%
Often	35%
Always	15%

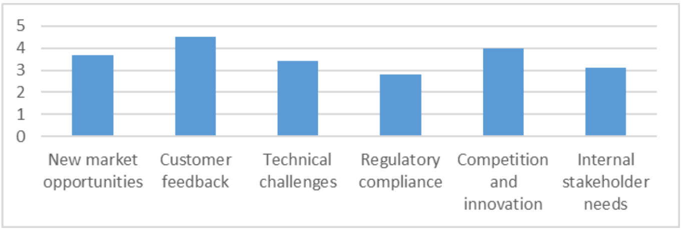

Figure 2. Reasons for Requirement Changes (Average Impact Rating)

Agile practitioners, such as product managers, programmers, business analysts, and other stakeholders, were polled in order to gather their feedback. The purpose of the survey was to collect information on the frequency and primary causes of requirement changes in initiatives. The significance of various factors was asked to be ranked by the participants from 1 (least important) to 5 (greatest importance). (most significant). The analysis of the survey data yielded tabulated results that demonstrated the most frequent triggers for requirement changes in Agile projects.

V. REQUIREMENTS PRIORITIZATION

The Agile methodology includes requirement prioritization, which enables teams to concentrate on completing the features or requirements that are the most essential first. This section outlines key steps in the requirement prioritization process, highlighting the importance of identifying stakeholders, defining user stories, prioritizing based on business value, technical feasibility, and risk, employing prioritization techniques, re-prioritizing regularly, and communicating priorities. In agile development, the main goal is to make the customer satisfied [19], if the needs of the consumers are met, the specifications for the system are thought to be finished (satisfaction of the customers). [24]. Agile teams can ensure that they are successfully prioritizing requirements and delivering value to consumers early and frequently if they adhere to these steps and follow them in the appropriate sequence. When selecting which requirements need to be implemented ahead of the others, a requirement prioritization [3] is crucial. However, it has been demonstrated that it is extremely difficult to prioritize requirements, with flexibility being one of the most significant challenges. [22] A significant number of stakeholders are typically involved in initiatives of this magnitude.

- Identifying Stakeholders: Identifying and involving all relevant stakeholders is the first step of the prioritization process. Customers, end-users, business analysts, product proprietors, developers, and other team members fall under this category. By involving all parties, a comprehensive understanding of the requirements is achieved, allowing for a more precise process of prioritization.
- Defining Requirements: Change Requirements in termed as user stories describe what a user intends to complete with the product. Each user story must have a distinct objective, scope, and acceptance criteria. These user stories serve as the foundation for prioritizing requirements, ensuring that the final product meets the demands and expectations of end-users.
- Prioritizing Requirements: The business value, technical feasibility, and risk of user stories must be prioritized. Priority should be given to the most important and valuable stories, followed by those with less business value. This strategy

assures that the team concentrates on customer-value-maximizing features.

• Employing Prioritization Techniques: MoSCoW, Analytic Hierarchical Process, Kano Model, and Value vs. Complexity are a few prioritization techniques that are accessible to Agile teams. To facilitate a more efficient prioritization process, teams should choose a technique that best fits their project and circumstance.

• Re-prioritizing: Due to priorities can shift at any point during the lifecycle of a project, it is essential to re-prioritize requirements on a frequent basis. The team should conduct a reassessment of their priorities at the conclusion of each sprint or iteration, considering any new information that has come to light along with comments from stakeholders. This ongoing reprioritization helps to ensure that the project continues to satisfy the ever-evolving requirements of the consumer.

• Communicating Priorities: Maintaining team unity and assuring emphasis on providing value to the client require open sharing of priorities. All team members ought to be conscious of the present priorities and comprehend the justification for the given significance hierarchy.

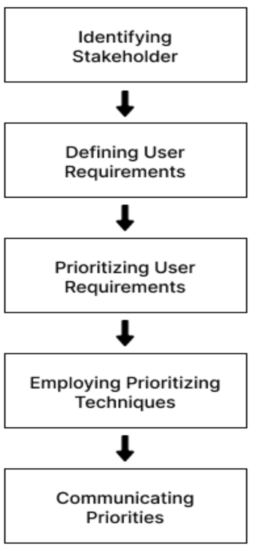
Identifying Stakeholder
Defining User Requirements
Prioritizing User Requirements
Employing Prioritizing Techniques
Communicating Priorities

Figure 3. Requirements Prioritization Workflow

How Sprint Backlog can be utilized in Requirement Prioritization

Product backlog is used as a receptacle for new requirements to substitute old requirements, repair problems, or eliminate functionality [21]. Product backlog is an acronym for "product requirements backlog." The "sprint backlog" is a limited collection of requirements that are contained within the "product backlog." These requirements are distributed among team members in preparation for an iterative development process known as "sprint" [21]. Within the context of the Agile methodology, a document known as the Sprint Backlog is a collection of tasks that have been chosen by the development team from the Product Backlog for fulfillment during the next sprint. During the course of the sprint, the tasks are segmented into more achievable subtasks and revised as new information becomes available.

It is expected to be concluded before the conclusion of the iteration, after which any tasks that have not been completed are added to the Sprint Backlog.

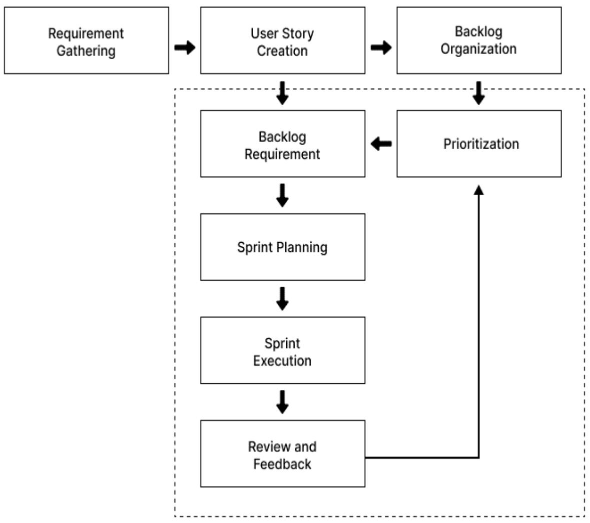

Figure 4. Workflow Model of Sprint Backlog in Agile

A Sprint backlog is a collection of work items (such as user stories, outstanding imperfections and other tasks) that is used by software teams to coordinate the work that needs to be done [20]. It is feasible to produce high-quality requirements if the requirements priority procedure is carried out properly [25].

TABLE IX. PHASE WISE DESCRIPTION OF SPRINT BACKLOG IN AGILE

Phase	Objectives	Benefits	Challenges
Requirement Gathering	Understand user standards and gather complete requirements.	Improves project success by ensuring the product surplus meets stakeholder needs.	Uncertain requirements, issues finding agreement among varied stakeholders.
User Story Creation	Translate requirements into user-centric, actionable work items	Promotes teamwork and user value.	Writing clear and concise user stories, defining accurate acceptance criteria.
Backlog Organization	Maintain a single, organized source of truth	Reduces confusion and repetition of work by facilitating	Maintaining the backlog, handling constraints, and managing links

	for all work items	easy prioritization and planning.	between user stories.
Prioritization	Determine the order of backlog items based on value, feasibility, and risk.	Ensures the team prioritizes the best features.	Balancing conflicting stakeholder priorities, accurately assessing value and risk.
Backlog Refinement	Continuously improve the clarity and readiness of backlog items	Improves sprint planning and execution, reduces unexpected issues.	Allocating sufficient time for refinement, managing scope creep.
Sprint Planning	Select and plan the work for the upcoming sprint.	Aligns the team around an aim and tasks, enabling resource allotment.	Accurately estimating capacity, constraints, and cross-team cooperation
Sprint Execution	Complete the selected user stories within the sprint timeframe.	Maintains client value while growing and improving.	Overcoming obstacles, motivating and collaborating.
Review and Feedback	Evaluate completed work and gather stakeholder feedback	Maintains stakeholder contact, improves and adapts to changing requirements	Obtaining honest and helpful criticism and promptly implementing it.

VI. ANALYTIC HIERARCHICAL PROCESS

The Analytic Hierarchy Process (AHP) is a systematic multi criteria decision making that was developed by Salty [1] for the purpose of handling complicated situations. If there are n total requirements, then each level of the hierarchy in which this technique is applied will conduct n(n-1)/2 comparisons [18]. In order to arrive at a conclusion, decision-makers need to take into consideration a number of different variables and make trade-offs.

Step 1: Define the hierarchy The AHP hierarchy consists of the overall goal, criteria, and alternatives (change requests in this case). Here, our goal is to prioritize change requests, and we have three criteria: Impact Analysis Score, Timeframe, and Effort Score. The alternatives are the seven change requests (CR-001 to CR-007). We define the hierarchy for the AHP implementation with the given data. Our goal is to prioritize change requests, which forms the top level of the hierarchy. The criteria and alternatives make up the next levels of the hierarchy. Here's an example of the AHP hierarchy for the given data: The hierarchy helps us to organize the complex problem of prioritizing change requests into a structured format. We have the overall goal at Level 1, the criteria that determine the priority at Level 2, and the alternatives (specific change requests) that we need to prioritize at Level 3.

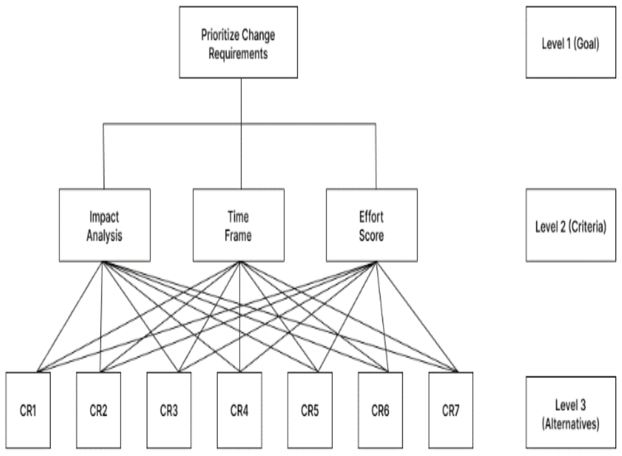

Figure 5. Hierarchical Structure of Prioritized Change Request

TABLE X. IMPLEMENTATION OF AHP ALGORITHM

Algorithm: Requirement Priortisation Using AHP

```
1. Define the hierarchy:
  1.1. Set Goal (objective) = "Prioritize Change Requests"
  1.2. Set Criteria = [Impact Analysis Score, Timeframe, Effort Score]
  1.3. Set Alternatives = [CR-001, CR-002, CR-003, CR-004, CR-006]
2. Create pairwise comparison matrix (A) and set equal importance for all criteria:
  2.1. Initialize matrix A with equal weights for all criteria
3. Calculate weight vector (w) from matrix A:
  3.1. For a consistent matrix with equal weights, set w = [1/3, 1/3, 1/3]
4. Create decision matrices for each criterion:
  4.1. For each criterion in Criteria:
    4.1.1. Create a decision matrix using the values of the criterion for each alternative
5. Normalize the decision matrices:
  5.1. For each decision matrix:
    5.1.1. Normalize the matrix by dividing each value in the matrix by the sum of the respective column
6. Calculate the weighted normalized decision matrices:
  6.1. For each normalized decision matrix:
    6.1.1. Multiply each value in the matrix by the corresponding weight from the weight vector (w)
7. Calculate the overall priority score:
  7.1. Initialize an empty list to store overall priority scores
  7.2. For each alternative:
    7.2.1. Calculate the overall priority score by summing the values of each row in the weighted normalized decision matrices
    7.2.2. Append the overall priority score to the list
8. Rank the change requests based on the overall priority score:
  8.1. Sort the list of overall priority scores in descending order
  8.2. Assign ranks to the change requests based on the sorted list
  8.3. Return the ranked list of change requests
```

Step 2: Create a pairwise comparison matrix for criteria as we assumed equal weights for all criteria (Impact Analysis Score, Timeframe, and Effort Score), the pairwise comparison matrix for criteria is:

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 5
DOI: https://doi.org/10.17762/ijritcc.v11i5.6589
***Article Received**: 20 February 2023 **Revised**: 27 March 2023 **Accepted**: 10 April 2023*

TABLE XI. PAIRWISE COMPARISON MATRIX

Criteria	Impact Analysis Score	Timeframe	Effort Score
Impact Analysis Score	1	1	1
Timeframe	1	1	1
Effort Score	1	1	1

Step 3: Calculate the criteria weights Since all criteria have equal importance, the weights for each criterion are equal (1/3). we calculate the criteria weights for the AHP implementation. In this specific case, we assume all criteria have equal importance, which means the weights for each criterion will be equal. Let's assign equal weights to the criteria: Impact Analysis Score (IAS), Timeframe (TF), Effort Score (ES). Since there are three criteria, we divide 1 (the total weight) by the number of criteria, which is 3: Weight for each criterion = 1 / 3 ≈ 0.3333. Now, we assign the calculated weight to each criterion: IAS: 0.3333, TF: 0.3333, ES: 0.3333. This means that each criterion has an equal influence on the prioritization of change requests in our AHP implementation. The weights will be used in subsequent steps to calculate the overall priority scores for each alternative (change request).

Step 4: Create decision matrices for each criterion for each criterion, we create a decision matrix using the given data.

TABLE XII. IMPACT ANALYSIS SCORE MATRIX

CR	Impact Analysis Score
CR-001	7.3
CR-002	6.8
CR-003	7.4
CR-004	7.2
CR-006	7.5

TABLE XIII. TIMEFRAME MATRIX

CR	Timeframe
CR-001	3
CR-002	1
CR-003	2
CR-004	4
CR-006	5

TABLE XIV. EFFORT SCORE MATRIX

CR	Effort Score
CR-001	6.0
CR-002	2.0
CR-003	5.0
CR-004	8.0
CR-006	10.0

Step 5: Normalize the decision matrices for each decision matrix, divide the values in each column by the sum of the column values. Here's how to normalize the decision matrices with the example data: First, calculate the sum of the column values for each decision matrix. Impact Analysis Score matrix sum: 36.2 Timeframe matrix sum: 15 Effort Score matrix sum: 31. Next, divide each value in each column by the sum of the column values.

TABLE XV. NORMALIZED IMPACT ANALYSIS SCORE MATRIX

CR	Impact Analysis Score
CR-001	7.3 / 36.2 = 0.201
CR-002	6.8 / 36.2 = 0.187
CR-003	7.4 / 36.2 = 0.204
CR-004	7.2 / 36.2 = 0.198
CR-006	7.5 / 36.2 = 0.207

TABLE XVI. NORMALISED TIMEFRAME MATRIX

CR	Timeframe
CR-001	3 / 15 = 0.2
CR-002	1 / 15 = 0.066
CR-003	2 / 15 = 0.133
CR-004	4 / 15 = 0.266
CR-006	5 / 15 = 0.333

TABLE XVII. NORMALISED EFFORT SCORE MATRIX

CR	Effort Score
CR-001	6 / 31 = 0.193
CR-002	2 / 31 = 0.064
CR-003	5 / 31 = 0.161
CR-004	8 / 31 = 0.258
CR-006	10 / 31 = 0.322

Step 6: Calculate the weighted normalized decision matrices Multiply each value in the normalized decision matrices by the corresponding weight (1/3 in this case). Here's how to calculate the weighted normalized decision matrices for the example data:

TABLE XVIII. WEIGHTED NORMALIZED IMPACT ANALYSIS SCORE MATRIX

CR	Impact Analysis Score
CR-001	0.201 * (1/3) = 0.067
CR-002	0.187 * (1/3) = 0.062
CR-003	0.204 * (1/3) = 0.068
CR-004	0.198 * (1/3) = 0.066
CR-006	0.207 * (1/3) = 0.069

TABLE XIX. WEIGHTED NORMALIZED TIMEFRAME SCORE MATRIX

CR	Timeframe
CR-001	0.2* (1/3) = 0.066
CR-002	0.066 * (1/3) = 0.022
CR-003	0.133 * (1/3) = 0.044
CR-004	0.266 * (1/3) = 0.088
CR-006	0.333 * (1/3) = 0.111

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 5
DOI: https://doi.org/10.17762/ijritcc.v11i5.6589
***Article Received**: 20 February 2023 **Revised**: 27 March 2023 **Accepted**: 10 April 2023*

TABLE XX. WEIGHTED NORMALIZED EFFORT SCORE MATRIX

CR	Effort Score
CR-001	0.193 * (1/3) = 0.064
CR-002	0.064 * (1/3) = 0.021
CR-003	0.161 * (1/3) = 0.053
CR-004	0.258 * (1/3) = 0.086
CR-006	0.322 * (1/3) = 0.107

Now, we have weighted normalized decision matrices for each criterion. The next step is to calculate the overall priority score by summing the values of each row in the weighted normalized decision matrices.

Step 7: Calculate the overall priority score Sum the values of each row in the weighted normalized decision matrices to get the overall priority score. For each change request, sum the corresponding values from the weighted normalized Impact Analysis Score matrix, Timeframe matrix, and Effort Score matrix.

TABLE XXI. OVERALL PRIORITY SCORE MATRIX

CR	Overall Priority Score
CR-001	0.067 + 0.066 + 0.064 = 0.197
CR-002	0.062 + 0.022 + 0.021 = 0.105
CR-003	0.068 + 0.044 + 0.053 = 0.165
CR-004	0.066 + 0.088 + 0.086 = 0.24
CR-006	0.069 + 0.111 + 0.107 = 0.287

Now, we have the overall priority score for each change request. The next step is to rank the change requests based on their priority score, from highest to lowest.

Step 8: Rank the change requests Sort the change requests based on the priority score, from highest to lowest. This was the AHP implementation for the given data. For a detailed calculation, please refer to the previous response where we went through each step with specific numbers. Here's how to rank the change requests for the example data:

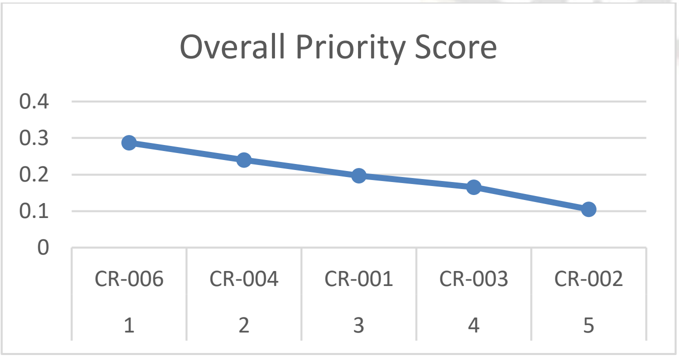

Figure 6. Overall Priority Score

VII. RESULTS AND DISCUSSION

In this study, we used the Analytic Hierarchy Process (AHP) to use the multi-criteria decision-making method to classify and rank change requests in Agile software development projects. Based on their Impact Analysis Score, time range, and effort score, the change requests were put into two groups: small and large. Here's how it's put together:

Small Change Request: CR-005, CR-007.

Large Change Request: CR-001, CR-002, CR-003, CR-004, CR-006.

Change Requests CR-001, CR-002, CR-003, CR-004, and CR-006 will go through the Requirement Prioritization process, whereas Change Requests CR-004 and CR-007 will go straight to the Implementation process.

After the Large Change Requests were put into groups, the AHP method was used to decide which ones were most important. This is how the list of priorities turned out:

1. CR-006
2. CR-004
3. CR-001
4. CR-003
5. CR-002

In the review of the results, it is emphasized how important it is for Agile projects to use a structured and objective way to group and rank change requests. The AHP method considers the fact that this process is subjective and complicated by using multiple factors and giving a clear, measurable reason for making decisions. The suggested method for separating Small Change Requests from Large Change Requests can help Agile teams handle their tasks and adapt to changes in the project's needs. By focusing on how to prioritize Large Change Requests, the project team can make the best use of its resources, improve the quality of the software, and shorten the time it takes to get the product to market. The study also shows how the AHP method could be changed to fit the needs of different Agile projects and groups. In conclusion, the results of using the AHP method to classify and rank change requests show that the multi-criteria decision-making technique works well and is useful for Agile software development projects. This method can help project teams deal with and adapt to changes better, which can lead to better software development results in the end.

VIII.CONCLUSION

As a powerful multi-criteria decision-making approach, the Analytic Hierarchy Process (AHP) was used in this study to categorize and rank change proposals in the setting of Agile software development initiatives. The Impact Analysis Score, the time time frame, and the effort score were used to categorize the proposals for change. Following that, the change proposals were divided into two groups: Small Change Requests: CR-005, CR-007. Large Change Requests: CR-001, CR-002, CR-003, CR-004, CR-006. The conclusions of this research highlight how important it is for Agile projects to employ a systematic and objective strategy to the categorization and prioritization of change requests. The AHP technique takes into consideration the intrinsic subjectivity and complexity associated with this process by incorporating multiple criteria. As a result, it provides a foundation for decision-making that is transparent, quantitative, and justifiable. Agile teams can improve their ability to successfully manage their responsibilities and adjust to changing project requirements with the assistance of the methodology that has been suggested for differentiating between minor and major change requests. Project teams have the ability to optimise resource distribution, improve software quality, and reduce the amount of time it takes to bring a product to market if they focus on the prioritization of major change requests. In addition, the research demonstrates the adaptability of the AHP technique, which can be reworked to meet the individual requirements and accommodate the unique conditions of a wide variety of Agile projects and teams. In conclusion, the successful implementation of the AHP technique to categorize and prioritize change requests provides evidence of its effectiveness and demonstrates that it is suitable for Agile software development projects. This technique has the potential to make change management and transition more effective, which will eventually result in better software development results.

ACKNOWLEDGMENT

This work is acknowledged under Integral University manuscript No. IU/R&D/2023-MCN0001930.

REFERENCES

[1] T. L. Saaty, ''Analytic hierarchy process,'' in Encyclopedia of Operations Research and Management Science. New York, NY, USA: Springer, 2013, pp. 52–64
[2] M. A. Akbar, Nasrullah, M. Shafiq, J. Ahmad, M. Mateen, and M. T. Riaz, ''AZ-model of software requirements change management in global software development,'' in Proc. Int. Conf. Comput., Electron. Electr. Eng. (ICE Cube), Nov. 2018, pp. 1–6.
[3] N. Nurmuliani, D. Zowghi, and S. Powell, ''Analysis of requirements volatility during software development life cycle,'' in Proc. Austral. Softw. Eng. Conf., 2004, pp. 28–37. 4. F. Sher, D. N. Jawawi, R. Mohamad, and M. I. Babar, "Requirements prioritization techniques and different aspects for prioritization a systematic literature review protocol," in Proceedings of 8th Malaysian Software Engineering Conference (MySEC). IEEE, 2014, pp. 31–36.
[4] R. Thakurta, "A framework for prioritization of quality requirements for inclusion in a software project," Software Quality Journal, vol. 21, no. 4, pp. 573–597, 2013.
[5] P. Achimugu, A. Selamat, and R. Ibrahim, "A web-based multi-criteria decision-making tool for software requirements prioritization," in Proceedings of International Conference on Computational Collective Intelligence. Springer, 2014, pp. 444–453.
[6] P. Achimugu, A. Selamat, and R. Ibrahim, "A preference weights model for prioritizing software requirements," in Proceedings of International Conference on Computational Collective Intelligence. Springer, 2014, pp. 30–39.
[7] J.K. Gershenson, and L.A. Stauffer, A Taxonomy for Design Requirements from Corporate Customers. Research in Engineering Design, 11 (1999),103–115.
[8] Hussain, S., Ehsan, N. and Nauman, S. (2010) A Strategic Framework for Requirements Change in Technical Projects: Case Study of a R & D Project. 2010 3rd IEEE International Conference on Computer Science and Information Technology (ICCSIT), 5, 354-35.
[9] Coram, M. & S. Bohner. The Impact of Agile Methods on Software Project Management. Engineering of Computer-Based System. ECBS '05. 12th IEEE International Conference and Workshops, 4-7 April 2005, p. 363-370 (2005).
[10] D. Baxter, J. Gao, K. Case et al., A framework to integrate design knowledge reuse and requirements management in engineering design. Robotics and Computer-Integrated Manufacturing, 24(2008), 585- 593.
[11] Cao, L. and Ramesh, B. (2008). Agile requirements engineering practices: An empirical study. IEEE software, 25(1):60–67.
[12] Albuquerque, D., Guimaraes, E., Perkusich, M., Costa, A., Dantas, E., Ramos, F., & Almeida, H. (2020, March). Defining agile requirements change management: a mapping study. In *Proceedings of the 35th Annual ACM Symposium on Applied Computing* (pp. 1421-1424)
[13] S. Ghosh, S. Ramaswamy, and R. P. Jetley, "Towards Requirements Change Decision Support," 2013 20th Asia-Pacific Softw. Eng. Conf, pp. 148- 155, Dec. 2013.
[14] Jyothi, V.E. & K.N. Rao. Effective Implementation of Agile Practices: Ingenious and Organized Theoretical Framework. International Journal of Advanced Computer Science and Applications 2 (3): 41-48 (2011).
[15] Minhas, N.M., Qurat-ul-Ain, Zafar-ul-Islam and Zulfiqar, A. (2014) An Improved Framework for Requirement Change Management in Global Software Development. Journal of Software Engineering and Applications, 7, 779-790.
[16] Quesf, A., 2010, Requirements Engineering in Agile Software Development, Journal of Emerging Technologies in Web Intelligence, Vol.2, No.3.
[17] P. Berander and A. Andrews, "Requirements prioritization," Engineering and managing software requirements, vol. 11, no. 1, pp. 79–101, 2005.
[18] Anwer, S., Wen, L., & Wang, Z. (2019). A systematic approach for identifying requirement change management challenges: Preliminary results. *Proceedings of the Evaluation and Assessment on Software Engineering*, 230-235.
[19] Sedano, T., Ralph, P., & Péraire, C. (2019, May). The product backlog. In *2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)* (pp. 200-211). IEEE.
[20] Alsalemi, A. M., & Yeoh, E. T. (2015, December). A survey on product backlog change management and requirement traceability

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 5
DOI: https://doi.org/10.17762/ijritcc.v11i5.6589
***Article Received**: 20 February 2023 **Revised**: 27 March 2023 **Accepted**: 10 April 2023*

in agile (Scrum). In *2015 9th Malaysian Software Engineering Conference (MySEC)* (pp. 189-194). IEEE.

[21] Asif, S. A., Masud, Z., Easmin, R., & Gias, A. U. (2017). SAFFRON: a semi-automated framework for software requirements prioritization. *arXiv preprint arXiv:1801.00354*.

[22] Zhang, Z., Li, X., & Liu, Z. L. (2014, September). A Closed-loop Based Framework for Design Requirement Management. In *ISPE CE* (pp. 444-453).

[23] Saeeda, H., Dong, J., Wang, Y., & Abid, M. A. (2020). A proposed framework for improved software requirements elicitation process in SCRUM: Implementation by a real-life Norway-based IT project. *Journal of Software: Evolution and Process*, *32*(7), e2247.

[24] AL-Ta'ani, R. H., & Razali, R. (2013). Prioritizing requirements in agile development: A conceptual framework. *Procedia Technology*, *11*, 733-739.

[25] Mueller, C., 2011, Requirements Management in an Agile-Scrum, Department of Computer Science San Marcos, TX.

[26] Shehzadi, Z., Azam, F., Anwar, M. W., & Qasim, I. (2019, August). A novel framework for change requirement management (CRM) in agile software development (ASD). In *Proceedings of the 9th International Conference on information communication and management* (pp. 22-26).

[27] LeClair, A., Bittner, K.: Agile Requirements Management, Forrester Research, 15 July 2016 20. 11th annual state of Agile™ survey, VERSIONONE.COM, VersionOne, Inc. (2016)

[28] I. Inayat, S. S. Salim, S. Marczak, M. Daneva, and S. Shamshirband. A systematic literature review on agile requirements engineering practices and challenges. Computers in Human Behavior, 51, Part B:915 – 929, 2015. Computing for Human Learning, Behaviour and Collaboration in the Social and Mobile Networks Era.

[29] S. McGee and D. Greer, "A software requirements change source taxonomy," in Software Engineering Advances, 2009. ICSEA'09. Fourth International Conference on, 2009, pp. 51-58.

[30] J. Buckley, T. Mens, M. Zenger, A. Rashid, and G. Kniesel, "Towards a taxonomy of software change," Journal of Software Maintenance and Evolution: Research and Practice, vol. 17, pp. 309-332, 2005.

[31] Glazer, H., Dalton, J., Anderson, D., Konrad, M.D., Shrum, S.: CMMI or agile: why not embrace both! (2008).

[32] Khan, A., Basri, S., Dominic, P., et al.: 'Communication risks and best practices in global software development during requirements change management: a systematic literature review protocol', Res. J. Appl. Sci., Eng. Technol., 2013, 6, p. 3514.

[33] B. M. M. Q. R. U. Q. M. A. Fateh ur Rehman, "Scrum Software Maintenance Model: Efficient Software Maintenance in Agile Methodology," in 2018 21st Saudi Computer Society National Computer Conference (NCC), Riyadh, Saudi Arabia, 2018.

[34] F. Moisiadis. A Framework for Prioritizing Software Requirements. PhD thesis, Macquarie University, Australia, July 2003.

[35] D. Summers, S. Joshi, and B. Morkos, ''Requirements evolution: Relating functional and non-functional requirement change on student project success,'' in Proc. 16th Int. Conf. Adv. Vehicle Technol., Aug. 2014, Paper DETC2014-35023, V003T04A002, doi: 10.1115/DETC2014-35023.

[36] S. Kugele, W. Haberl, M. Tautschnig, and M. Wechs, ''Optimizing auto matic deployment using non-functional requirement annotations,'' in Proc. Int. Symp. Leveraging Appl. Formal Methods, Verification Validation. Berlin, Germany: Springer, 2008, pp. 400–414.