Computers, Materials & Continua

Tech Science Press

DOI: 10.32604/cmc.2024.053830

ARTICLE

Towards Improving the Quality of Requirement and Testing Process in Agile Software Development: An Empirical Study

Irum Ilays[1], Yaser Hafeez[1,*], Nabil Almashfi[2], Sadia Ali[1], Mamoona Humayun[3,*], Muhammad Aqib[1] and Ghadah Alwakid[4]

[1]University Institute of Information Technology, Pir Mehr Ali Shah Arid Agriculture University, Rawalpindi, 46000, Pakistan

[2]Department of Software Engineering, College of Computer and Information Sciences, Jouf University, Al Jouf, 72388, Saudi Arabia

[3]School of Arts Humanities and Social Sciences, University of Roehampton, London, SW15 5PJ, UK

[4]Department of Computer Science, College of Computer and Information Sciences, Jouf University, Al Jouf, 72388, Saudi Arabia

*Corresponding Authors: Yaser Hafeez. Email: yasir@uaar.edu.pk; Mamoona Humayun. Email: Mamoona.Humayun@roehampton.ac.uk

Received: 10 May 2024 Accepted: 26 July 2024 Published: 12 September 2024

ABSTRACT

Software testing is a critical phase due to misconceptions about ambiguities in the requirements during specification, which affect the testing process. Therefore, it is difficult to identify all faults in software. As requirement changes continuously, it increases the irrelevancy and redundancy during testing. Due to these challenges; fault detection capability decreases and there arises a need to improve the testing process, which is based on changes in requirements specification. In this research, we have developed a model to resolve testing challenges through requirement prioritization and prediction in an agile-based environment. The research objective is to identify the most relevant and meaningful requirements through semantic analysis for correct change analysis. Then compute the similarity of requirements through case-based reasoning, which predicted the requirements for reuse and restricted to error-based requirements. Afterward, the apriori algorithm mapped out requirement frequency to select relevant test cases based on frequently reused or not reused test cases to increase the fault detection rate. Furthermore, the proposed model was evaluated by conducting experiments. The results showed that requirement redundancy and irrelevancy improved due to semantic analysis, which correctly predicted the requirements, increasing the fault detection rate and resulting in high user satisfaction. The predicted requirements are mapped into test cases, increasing the fault detection rate after changes to achieve higher user satisfaction. Therefore, the model improves the redundancy and irrelevancy of requirements by more than 90% compared to other clustering methods and the analytical hierarchical process, achieving an 80% fault detection rate at an earlier stage. Hence, it provides guidelines for practitioners and researchers in the modern era. In the future, we will provide the working prototype of this model for proof of concept.

KEYWORDS

Requirement prediction; software testing; agile software development; semantic analysis; case-based reasoning

1 Introduction

In software engineering (SE), agile is an iterative paradigm for developing software and project management which is helpful for teams to accomplish customer satisfaction in this continuously evolving period [1,2]. Agile-based environment (ABE) is a project management methodology emphasizing flexibility, collaboration, and continuous improvement [3,4]. In the modern era, an ABE allows information technology organizations to stay competitive by continuously improving their products and services to meet changing customer needs and market trends [1,3]. The two major phases that are crucial for software development and the success of any project are requirements specification and testing [2,5]. Gathering and defining the functionality of developed software during requirement engineering is a problematic stage of the developed software; because it involves ambiguous requirements, multiple perspectives from stakeholders, irrelevant requirements, validations of changes, and coordination between the team and stakeholders in ABE [1,4,5]. As a result, it makes the software more complex and reduces its productivity and quality [2]. The goal of requirement prioritization (RP) is to ensure that the software project's needs which are of the utmost importance are recorded and carried out first [5]. Requirement dependencies and inconsistencies may be mitigated by careful prioritization [5]. Different approaches such as clustering, and analytical hierarchical processes (AHP) have been used to specify and prioritize requirements based on stakeholders, budget, timeline, and the kind of project [6–8]. The clustering approach separates a given collection of data into numerous clusters to establish the proximity between those items, whereas, AHP prioritizes the requirements based on significance cost, time, penalty and risk. The AHP is used by the stakeholders to rate the needs. AHP is a technique for structuring and deriving conclusions from challenging mathematical decision-based situations. AHP pairs requirements according to importance (based on stakeholder perspectives), consequences (according to the severity of the effect, if executed incorrectly), cost (tested requirements in accordance with business, implementation, and maintenance cost), time (implementation, deployment, maintenance and response time), and risk (relevant requirements in accordance to the quality, correctness, completeness, and other factors) [9,10].

In case-based reasoning (CBR), a system learns from previous experiences or cases and applies that knowledge to solve new problems [8]. The cases are stored in a knowledge repository and are evaluated to identify the most similar situation to the current one. Once a match is made, the system will use previous experience to solve the existing problem [11]. RP has major issues which are redundancy and irrelevancy of requirements due to multiple viewpoints of stakeholders. This increases the cost of the system, and decreases the quality and low satisfaction level, so there is a need to improve these parameters. Semantic analysis (SA) analyzes text documents to extract meaningful and relevant information. It involves using natural language processing techniques to identify and understand the underlying meaning of words and sentences [6,12]. Text mining is extracting valuable insights and knowledge from unstructured text data. Text mining techniques can be used to uncover hidden trends, patterns, and relationships in large volumes of textual information [7,8].

In this research, we are using SA using text mining to improve the requirements specification. The requirement classification and clustering are used to improve the irrelevancy and redundancy of requirements. Therefore, we are using CBR to map on whether requirements are reused, new or updated.

Software testing is a substantial phase in software development that ensures software quality by testing a program against its target and identifying software faults [13]. Therefore, trust testing during application development to validate software reliability and quality after implementation of changes [14]. Testing has major threats regarding fault detection and needs to reduce the redundant

and irrelevant test cases during ABE. Test case prioritization (TCP) uses pre-established goals to prioritize the test cases. In contrast, test case selection (TCS) focuses on the crucial test cases impacted by the changed area [3,15,16]. TCP got more attention since test cases are included in the test suite [17]. Furthermore, if the process is interrupted, TCP helps reduce the time required for retesting [18,19]. It encourages us to add the TCS process, with time and budget limitations, eliminates the redundancy of test cases from the prioritized test suite [19]. Moreover, TCP schedules the test cases using various test adequacy criteria, such as optimizing requirements coverage [20]. The testing criteria significantly influence the approach's efficiency [20]. Different standards, like cost or coverage optimization, are used to evaluate the TCP. It is not easy to assess the relevance, interrelation, and relative value of these aspects [20]. In TCP, a large number of test cases are involved for execution to validate the changes and most of the test cases are reused for execution to identify the faults [21]. Therefore, there is a need to mine test cases based on different associations and relationships to improve the decision-making process. Thus, apriori algorithm is the data mining technique to deal with large datasets based on association rules. It helps in TCP by analyzing different patterns and relationships among test cases that help in validating the changes [22]. In this research, we improved the TCS and TCP by correct specification, reuse and restructuring after modification in requirements. We identified the error-based requirement through apriori algorithm during the requirement prediction phase, which also improves the test cases irrelevancy, redundancy and fault detection. The apriori method is used by the model to examine the impact of modifications. It uses a map of requirement frequency to select suitable test cases. Test cases are then prioritized depending on their frequency of repeat. This method focuses on frequently reused test cases to improve problem identification, once modifications are implemented. These examples are likely to expose potential faults produced by the changes, enhancing overall software quality and testing efficiency.

1.1 Research Problem

The purpose of this study is to investigate the impact of various variables on the TCP in ABE. The following research questions (RQs) (RQs are based on the research process) are being addressed:

RQ1: Do requirements affect TCP?

The question in this study looks into the relationship between TCP and requirement specification by examining how more explicit and extensive requirements result in better TCP.

RQ2: How does the proper reuse and selection of requirements affect the TCP procedure?

The purpose of the question is to investigate the effect of reusing and selecting relevant requirements on the efficiency and effectiveness of the TCP process. It may be able to reduce redundancy and improve overall testing quality by doing so.

RQ3: Does the change impact analysis of the proposed model result in an improvement to the TCP?

The RPTSP (Requirement Prediction, Test-case Selection, and Prioritization) potentially increases the adaptability of the TCP process to change by including a change impact analysis using the apriori algorithm to identify frequently changing requirements and prioritize associated test cases.

RQ4: Can proposed RPTSP improve fault detection rate (FDR)?

The current study evaluates the efficiency of RPTSP, in improving the rate of fault detection through improved requirement prediction, TCS, and prioritization. The average percentage fault detection (APFD) metric is used in the study to assess the model's usefulness in detecting software faults.

The goal of TCP procedures and addressed research concerns in ABE is to develop higher quality software in the field of software development by optimizing increased FDR. An intense amount of research has been accomplished to find ways to efficiently predict the requirements and increase FDR. On the other hand, existing methods such as clustering and AHP in ABE cannot efficiently and effectively predict requirements and identify faults after changes. The redundant and irrelevant requirements result in redundant faults identification and selection of test cases. The APFD metric percentage is used to find FDR over the prioritized test suite. It detects the maximum faults, which results in a higher FDR.

1.2 Research Contributions

The main research contributions to address the difficulties described above in the following ways:

- In this research, we present a requirement prediction, TCS, and prioritization method for agile-based software development based on irrelevant and redundant TCS to enhance the ability of FDR.
- The proposed model deals with requirement prediction of similar and frequent changes in requirements, selection of test cases, and prioritization of frequently changed test cases in every release for agile-based development strategies. Therefore, our proposed model named as RPTSP. In the first phase, we classify requirements to reduce ambiguity among stakeholders; we categorize the requirement through SA. In clustering, the CBR technique identifies the similarity between requirements to remove the relevancy and redundancy between the requirements. Afterwards, parameters are generated to select and prioritize test cases to identify maximum faults. TCS and prioritization improve the FDR through APFD.
- The RPTSP model was evaluated using case studies and experiments in three different projects. After observing different results from these projects, it was discovered that the RPTSP model identifies the most accurate requirements, detects more faults, and is effective for all the projects compared to AHP, clustering, and other FDR methods.

1.3 Research Significance and Applicability

This research provides guidelines and a baseline for researchers, requirement engineers, and testers to enhance the quality and productivity of the software. This research significantly improves the activities of software engineers and testers, such as correct reuse, requirement selection, and validation during requirement specification, as well as testing after implementing changes in ABE.

The remaining paper is categorized into different sections as follows: Section 2 provides an overview of the related work. Section 3 provides the details of the RPTSP model, followed by Section 4, which provides the results and discussion, Section 5 is about practical implications, Section 6 concludes the paper and finally Section 7 is about potential limitation and future work.

2 Related Work

Software keeps changing constantly through development and maintenance for various reasons, like addition of new features, modifying existing components, and refactoring [6]. Software testing plays a crucial role in ensuring the accuracy, completeness, and utility of user specifications and requirements. On the other hand, testing is a process that ensures the software is defect-free or failure-free. Prioritization of test cases is a difficult task in software testing. In contrast, TCP tries to determine the best order for test case executions to maximize the testers' work, even if testing is stopped early

for any reason; like when the testing resources are depleted [23]. Software requirements and software testing procedures need to be aligned, to prevent possible issues with software product delivery. For instance, inadequate communication with testers regarding requirements modifications may lead to an incorrect verification of outdated requirements or a failure to verify new requirements, which could compromise the quality of the software. Thus, one of the key elements that determines the project's success and, ultimately, the level of customer satisfaction is the quality of the software. But making sure software is of a high caliber is no easy feat. Teams working on software development have been concentrating on creating products that satisfy customers in the face of constant change and shortened deadlines [24].

Different issues of the RP and testing are highlighted in different studies. Software testing and RP subjects in SE are significant areas of research. There may be various explanations, but one possibility is that the variables aren't considered crucial in isolated studies. Therefore, alternative prioritization strategies must be compared and examined across numerous platforms (e.g., time and cost) to obtain credible evidence. In paper [25], a search-based SE method was used for the ranking and selection of requirements. The goal is to rank and pick the prioritized requirements in software. This method gives an evaluation and relationship detected, examined, categorized, and grouped search-based SE approaches that were put forward for problems with need selection and ranking. According to [26], they offered the idea of a multi-criteria decision-making model for RP. Their technique is a case study that uses a neural network-based model and fuzzy AHP to identify the best stakeholders while ensuring the stakeholder's satisfaction [10,27]. Few of the prioritization techniques in the literature satisfy particular quality standards, including effectiveness, ease of use, and scalability [9,10]. The paper [25] suggested a technique that involved end-user participation in a situation-transition framework for RP. In the paper [28], the authors considered the multi-users viewpoints and proposed a commercial off-the-shelf (COTS) prioritization technique. In [17], the authors provided a systematic mapping of literature to categorize current approaches to solve the problems like selection and prioritization of requirements.

Similarly, the paper [29] suggested a framework for prioritization: a fuzzy-based engine (FBE). In the FBE approach, Fuzzy rules are used to assign a user prioritization value as input to the requirement analysis process. Paper [30] proposed a technique to deal with new and current requirement priority orders through machine learning. The RP is based on users' feedback with less human effort to reduce cost and time [31]. Paper [32] suggested a combination of evolutionary-based and clustering procedures to handle huge data effectively using ranks. Therefore, literature identifies different problems for RP: scalability, time consumption, accuracy, etc. The method has been used to evaluate trace link assurance by looking at similar and different requirements to reduce complexity. SA using text mining combines both these techniques to analyze large volumes of text data and extract valuable insights.

As we know, the latest trends are moving toward AI, so there is a need to integrate AI in RE and testing [21]. The apriori algorithm is a data mining technique to discover association rules in large datasets. It is primarily used to analyze transactional data such as that found in retail sales or customer transaction history. Primarily, the apriori algorithm works on a database encompassing a considerable number of transactions [22]. Essentially, the apriori algorithm identifies frequent item sets in the data, i.e., groups of items frequently occurring together in the dataset [22]. In [33], the authors has used a prioritization model developed on a different dimensional equation of TCP for early execution to increase flexibility in the TCP process. They also discovered that considering weighted probability distributions might enhance the effectiveness of dynamic TCP techniques. The technique's limitation is that it fails to keep track of test case repositories after modifications and overlooks test changes and historical fault information. Regarding FDR, historical data with code

coverage is a more effective method for tracking test cases concealing modified lines [15]. The disparity metric is a highly straightforward, effective, and efficient method of TCP. Adaptive random testing is directly applied in this method of dispersity metric, which is not a revolutionary concept in and of itself. Hence, there is a need to identify the redundant fault detection and change in requirements at any level to accommodate [34].

A method is proposed for TCP following the pattern extraction of neurons and their values from the training set above the deep neural network in the paper [13]. The TCP across a deep neural network model is determined based on how well activated and inhibited neurons adhere to recognized patterns [13]. The initial test case priority was to determine the prioritization technique using historical data and prioritized requirements [23]. Despite this approach's adequate performance, redundant faults were found, when faults were divided according to requirement property. Additionally, multiple prioritization criteria were used, but the frequency of test case changes was overlooked.

According to the existing studies, we determined that integration of requirements in testing is challenging and prone to error tasks. This is because of the lack of term mismatch and semantics resulting from different stakeholder views during the formulation of requirement specifications. After the changes, inconsistency, human collaboration, ambiguity, and uncertainty in real-time systems increased. In this research, we identified different parameters from the existing literature, that are the main gaps of testing based on requirements in an ABE; we identified incorrect or irrelevant requirement prediction, test cases irrelevancy, and redundant fault detection after changes in implementation as depicted in Table 1. The identified challenges (such as accurate selection of requirements, stakeholder satisfaction and term mismatch) in ABE are due to continuous evolution or changes in the software development process. The main reason for these challenges in ABE is due to lack of procedure, lack of structured planning and organizing meetings, complexity in adopting new methodology by the team and effort management. These challenges increase the errors and reduce FDR during the validation of changes in ABE. Therefore, our proposed model RPTSP improves the FDR by addressing existing challenges using semantic analysis and CBR technique for the accurate TCP. These challenges are mentioned in different existing studies and highlighted that it increases the error and reduces FDR during validation and changes in ABE.

3 Proposed RPTSP Model

The proposed RPTSP model was developed through a structured and systematic approach that involved the collection of data and knowledge from various sources, as well as the collaboration of multiple stakeholders and experts. To ensure efficiency, accuracy, and overall quality of the software product, an integrated approach for software requirements and testing these two main stages of software development are aligned specifically. The main scope of our proposed model is to improve the software testing process in agile environment by aligning requirements with the test cases instead of improving agile methodology. The RPTSP efficiently and effectively predict requirements and identify faults after changes. The redundant and irrelevant requirements result in redundant faults identification and selection of test cases. The APFD metric percentage is used to find FDR over the prioritized test suite. It detects the maximum faults, which results in a higher FDR. The primary goals of this RPTSP model are shown in Fig. 1, which are to enhance the identification of duplicate defects after modifications and inaccurate requirement prediction. SA is used to categorize user-based needs and system-based requirements. Utilize CBR to cluster the needs after categorization. Predicting the needs based on high priority eliminates the ambiguity in situations when priorities are comparable,

and picking the highest priority requirements from each cluster allows for the fastest possible mistake detection. The three stages of the RPTSP model are defined below:

Table 1: Comparison of existing studies

Parameters\References	[6]	[19]	[13]	[21]	[23]	[34]	[30]
Accurate selection of requirements	■□	■□	■□	■□	□□	■□	■■
Change in requirements	■□	■□	■□	□□	□□	□□	■■
Redundancy	□□	□□	□□	□□	■■	■□	□□
Irrelevant requirements	□□	■□	□□	□□	■□	□□	□□
Stakeholder satisfaction	■■	□□	□□	■□	■□	■□	■■
Semantic analyses	■■	□□	■□	□□	■□	■□	□□
Lack of AI method	■□	□□	■■	■■	■□	□□	■■
Term mismatch	■□	□□	□□	□□	■□	■□	■□
Fault detection	□□	■■	■■	□□	■■	□□	■■
Irrelevant test cases	□□	■□	□□	□□	□□	■□	□□
Change in test cases	■■	■□	□□	■□	□□	■□	■■

Note: Not mentioned = □□, Partially mentioned = ■□, Mentioned = ■■.

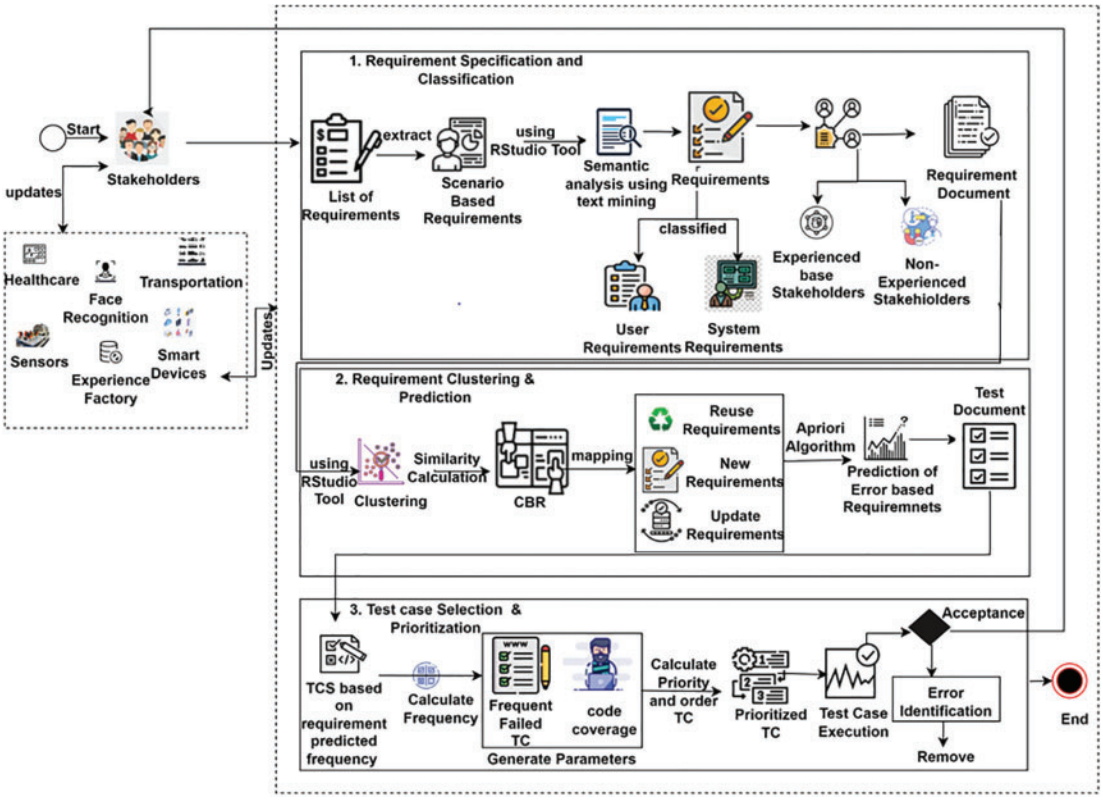

Figure 1: RPTSP model

The first phase of the model is the requirement specification and classification. This phase comprises of the following main stages:

a) Scenario-based requirement: Scenario-based modeling identifies the primary use cases for the proposed software. Scenarios are instances of experience with a system captured by users. A scenario-based requirement explains the user expectation from the system and represents the business value that the cross-functional team should make the product in a sprint. For the selected sprint, each user scenario is kept in a file. Every user story has multiple requirements. For example, the car alarm system is a scenario and most common user requirements or expectations are door lock, unlock, alarm activation and deactivation. It helps in the specification of requirements.
b) Classification of requirements: In RPTSP model, the requirements are classified into user-based and system-based requirements through SA. SA mine requirements are based on stakeholders' priority and remove conflicts using the RStudio tool for text mining (this tool extracts text semantically from all the documents). The requirements are extracted through the experience of stakeholders; if the stakeholder has more than one year of experience, they are labeled as experienced users and they worked as a technical user like requirement engineer, developer and tester. These users emphasize the technical feasibility, integration, and detailed functionality of the requirements. They prioritize aspects that ensure robust system performance, security, and technical integration. Otherwise, they are labeled as non-experienced users and they worked as a non-technical user like end-user, product owner and customer. These users prioritize ease of use, intuitive interfaces, and overall user experience. They focus on functionalities that directly impact their interaction with the vehicle. The impact assessment was done on the basis of high impact, medium impact and low-impact requirements. The high impact requirements that significantly affect the system's usability, security, and user satisfaction. The medium impact requirements that improve user experience but are not critical for system operation. Then low impact requirements that provide additional convenience but have minimal impact on core functionalities.

The second phase is about requirement clustering and prediction process:

Cluster: In RPTSP model, the cluster module represents the clustering of requirements to identify the similarity through the CBR method. CBR is used for the feature's reusability and stakeholder's involvement. This method provides priority for those requirements that are similar and relevant to improve the prediction process. CBR utilizes a prior, comparable solution's insight to rank the requirement for scoring new functionalities and save position into the repository [7]. The CBR steps are:

- Retrieval of requirements: By using expert knowledge, we compare prior similar requirements with the same functionalities, and the ranking of current functionality is stored in the repository.
- Reuse ranking: To ensure similar requirements through requirement interaction in the prediction phase, we reuse the previous ranking of stakeholders.

The RPTSP model also works with a semi-supervised clustering method called semi-supervised K-means. The overall number of clusters in the K-mean clustering method is determined by the value K, which is the number of items to arrange requirements in clusters. Fig. 2 shows the procedure, which begins with take-out data information using value K and ends with clustering data based on similarity. Finally, we obtain several clusters of massive unformed data to decrease the ambiguity and complexity.

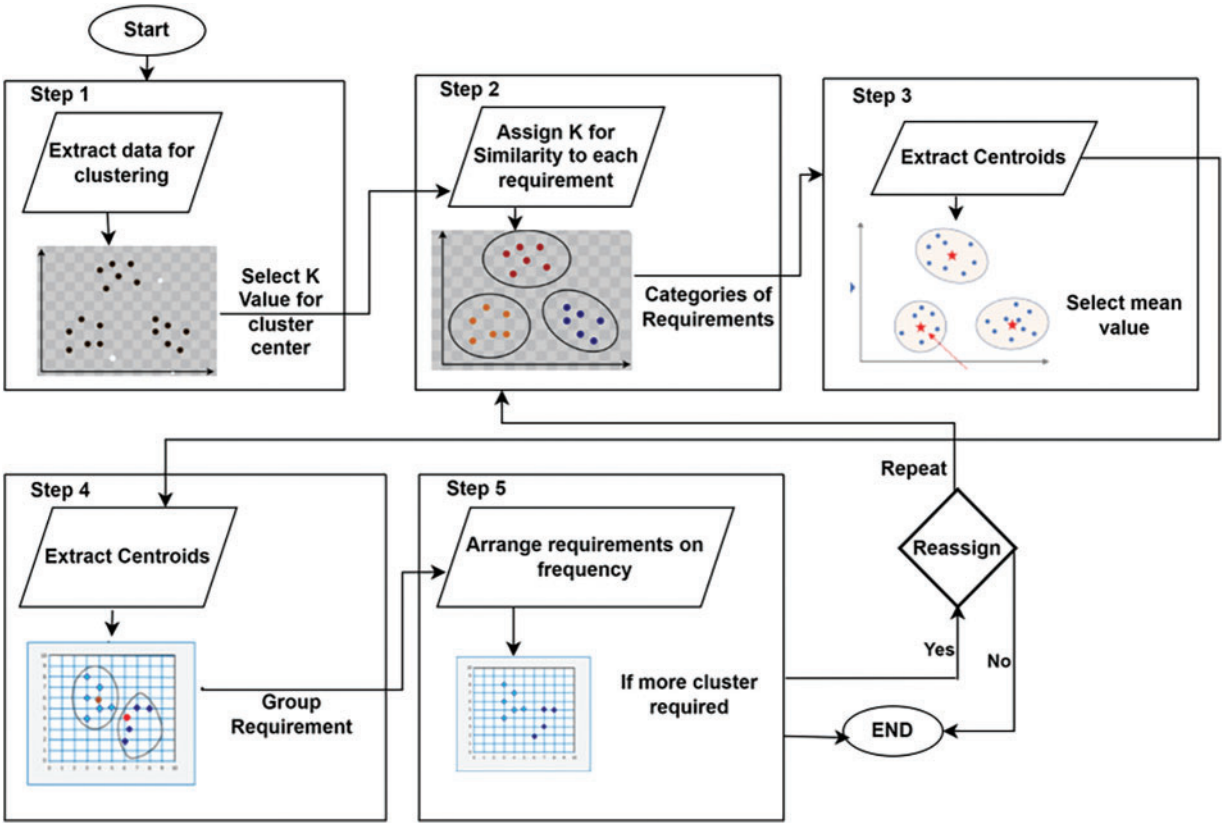

Figure 2: Clustering of requirements

Clustering similar requirements and reusing previously ranked functionalities minimize the redundancy in requirement documentation. Categorizing and clustering requirements early in the process allows for a more streamlined requirement gathering, reducing the time and effort needed to identify and document all requirements. Clustering requirements in agile methodologies, as demonstrated by the RPTSP model, significantly enhances the efficiency and effectiveness of both requirement gathering and testing processes. This integrated approach aligns requirements with test cases, reduces redundancy, improves fault detection, and optimizes resource allocation, leading to higher overall product quality and stakeholder satisfaction.

Prediction: After clustering of requirements, we used CBR to map the requirement for reuse, new and update based on similar requirements to satisfy stakeholders' needs. After that, we apply the apriori algorithm to identify frequent patterns in these similar requirements to decrease irrelevancy and ambiguities during the collection of relevant test cases based on change requirement. Like if we extract requirements R2 and R3 as a reuse requirement, R4 as a new requirement and R5, R6 for modification. Then we identified R2, R3 are frequently used and R5, R6 are updated frequently using apriori algorithm for TCS.

For prediction of requirements, we consider both requirements that is new and change to improve the testing quality process. The prediction process used to increase relevancy for the selection of test cases to correct prioritization of test cases to validate the new requirements and change requirements.

The last phase about TCS and prioritization of each requirement is done once the requirements have been clustered and predicted. After that, the requirement selection process is formalized to contain all the test cases chosen from all the extracted requirements. Before this step, redundancy and irrelevancy were tried to be removed, and test cases were generated on requirement prediction criteria to improve the FDR after changes. After that, test cases are ranked and prioritized to guarantee that the most crucial are chosen first. The APFD is used to calculate the maximum faults to improve the FDR.

Each stage of the RPTSP model uses data-driven techniques (SA, CBR, semi-supervised K-means, and apriori algorithm) to justify the ranking of requirements. These methods ensure that rankings are based on actual usage patterns, historical data, and stakeholder priorities, rather than subjective judgment. The RPTSP model emphasizes removing redundant and irrelevant requirements, which reduces the likelihood of redundant faults and ensures that test cases are aligned with the most critical requirements.

4 Results and Discussions

In this section, we investigate the effectiveness and performance of the RPTSP model by applying three projects or cases for the evaluation of the proposed model (PM) using experiments. The reason for selecting these cases is based on their backgrounds, scope, and domain. The first case is A, about the car security system, the second case is B, a patient-centric health system named iTrust, an open-source system, and the third case is C, about a web-based IT application selected from a real-world industrial system. All cases belong to the same organization and the same number of participants during the implementation of these three cases during conducting the experiment. Due to confidential reasons, the company's links are not mentioned here, and names are substituted by some other general name in Table 2. These software companies used distinct methodologies aimed at classification and prediction for higher user satisfactions levels and to increase human interaction productivity. Table 2 describes the specification of all three cases. The software's which were used for experimental setups are; RStudio used for semantic analysis, clustering and apriori algorithm, JUnit used for test case execution, SPSS 23 for statistical analysis, drawio used for drawing models, Origin used for graphs, and window 10 used as operating environment.

Table 2: Specification of projects

Projects	No. of use cases	No. of software versions	Size (line of code)	No. of faults	No. of test cases
Case A	653	5	3532	33	960
Case B	24,018	6	276,456	45	1334
Case C	15,541	7	155,987	40	1420

The evaluation process is based on experience, applicants' knowledge, and related to these projects. Applicants of the organization decided to implement the RPTSP model for user satisfaction and product quality. We selected 12 participants and evenly split them into the Experiment Group (EG) and Control Group (CG). In each case, we selected different participants with the same number of participants. We implemented 20 use cases in each case for all the selected cases during the experiment due to restrictions applied by the selected organization. The EG group developed using RPTSP, and the CG group used the traditional method which is AHP and clustering for all projects and both groups work in ABE. The project participants are Project manager owner (PMO), Developers

(Ds), Requirement analyst (RA), Quality analyst (QA), Stakeholder (Sr), and Team leaders (TL). After implementation of projects, examine the progress and evaluate through some parameters (as depicted in Table 3), identified from the existing literature to improve requirement prediction, TCS and prioritization.

Table 3: Parameters for evaluation

Parameters	Abbreviation	Parameters	Abbreviation
Requirement identification and retrieval	RIR	Enhance requirement integration	ERI
Proactive to changes	PTC	Completeness of requirements increased	CRI
Easy to adopt	EA	Process accuracy increase	PAI
Requirement prediction process	RPP	Requirements redundancy remove	RRR
User satisfaction increased	USI	Semantic analysis	SA
Increase productivity	IP	Human interaction reduced	HIR
Term mismatch resolves	TMR	Formal specification	FS
Requirements conflict remove	RCR	Fault detection rate	FDR
Predicted desirable requirements	PDR	Frequently change test cases	FCTC

The efficiency of the RPTSP model is evaluated in terms of practical calculation which is mainly focused on the following experiment questions (EQs) (EQs for the approval of PM effectiveness and performance during experiment):

- **EQ-1:** What impact does a semantic-based requirements classification and prediction have on the outcome of the requirement specification process?
- **EQ-2:** Are RPTSP implementation outcomes better than other relevant methodologies?
- **EQ-3:** Can RPTSP effectiveness improve the relevance of the requirements integration process?
- **EQ-4:** Does RPTSP improve the FDR through the requirement prediction method?

To answer EQs based on some parameters that are shown in Table 3 to evaluate the satisfaction level of the participants, the results are depicted in Fig. 3–5, and the results showed more increases in satisfaction level parameters. Table 3 mentions the identifies parameters from the literature and reason to select these parameters and their impact on the process of change validation during TCP [3,6,11,19–23,30,34]. All these parameters are based on the increase in FDR after the TCP using frequently used test cases or not frequently reused to validate the changes. Therefore, improvement in the TCP process increases the FDR which impacts on the satisfaction level of the participants.

Hence, the first step of the experiment starts by extracting requirements through SA using text mining from experiment participants and then mapping with mentioned requirements. These requirements were classified as system-based and user-based requirements and then documented. The experiment started from the first step, in which we extracted requirements with their complete constraints and stakeholder viewpoints. RStudio was used to extract terms from all projects. For example, in the car security project, some of the terms extracted after the text mining process are listed in Table 4.

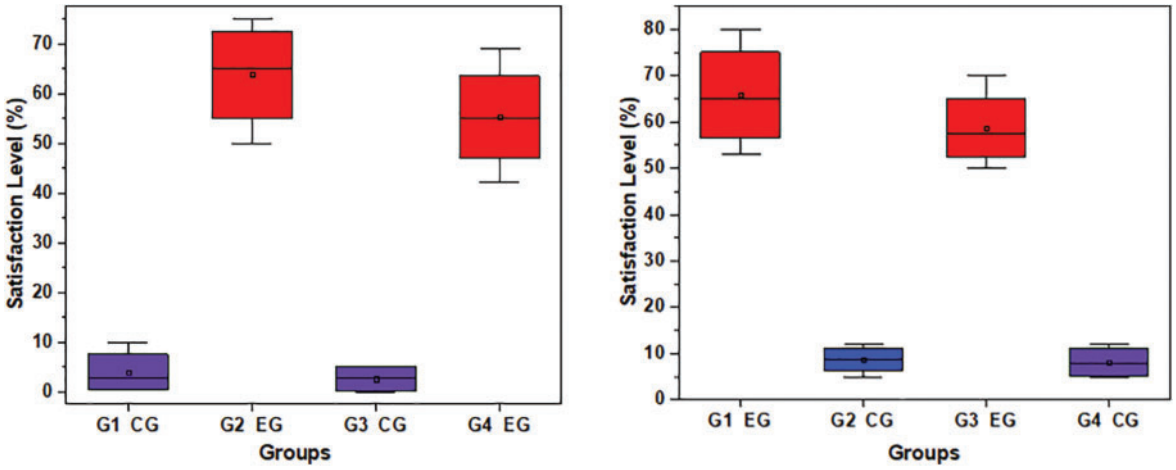

Figure 3: Factor analysis/satisfaction level (highly satisfied and dissatisfied)

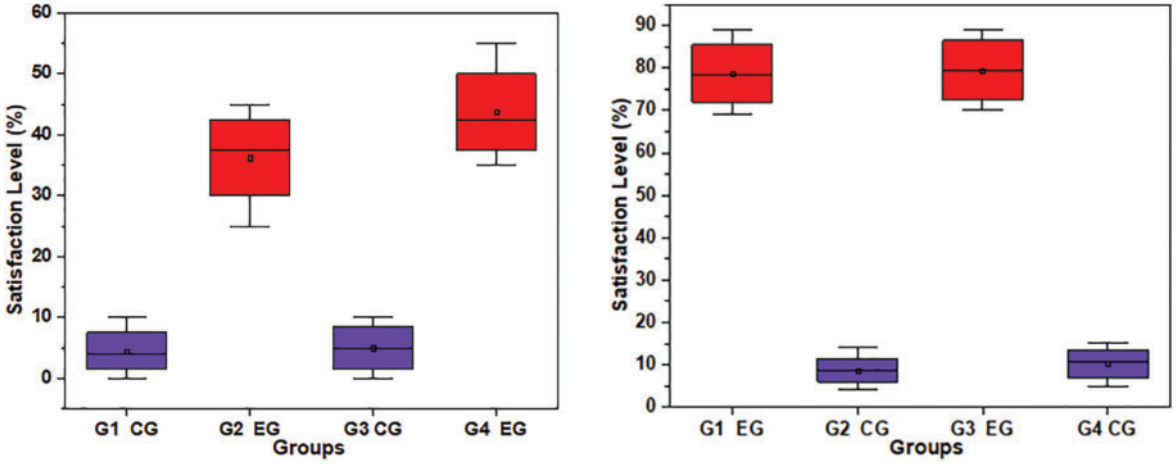

Figure 4: Factor analysis/satisfaction level (satisfied and neutral)

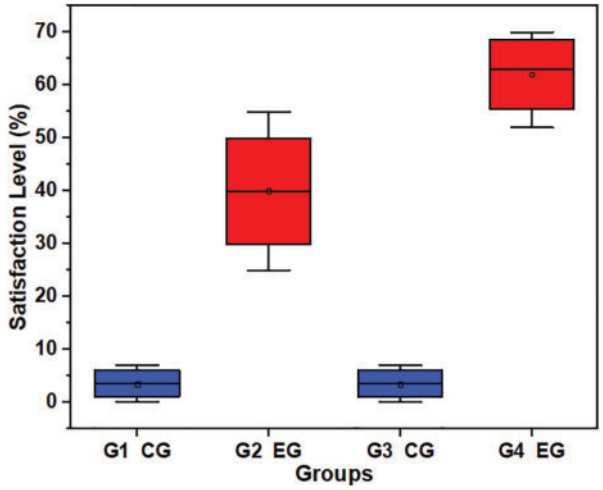

Figure 5: Factor analysis/satisfaction level (highly dissatisfied)

Table 4: Requirement extraction through text mining for case A (car security system)

Sr. #	ID	Functionalities of requirements	Detailed examples and justifications
1	R-1	Unlock door	Example: A user can unlock car door by using a remote key from distance for quickly and conveniently. Justification: There are three methods used to unlock car door remote key, manual key and smart system. This is important access and security.
2	R-2	Light blink	Example: A user can lock or unlock the car by using remote key and flash lights blink shortly. Justification: This ensures the user that car status is change.
3	R-3	Alarm activate	Example: The alarm of the car starts with loud voice and lights blinks. When unauthorized user is trying to access the car like broken window. Justification: This ensures the car safety from any theft attempt and informed user for any unusual activity.
4	R-4	Alarm deactivate	Example: The car user deactivates the alarm and lights by accessing remote key. Justification: For maintenance of security and any disturbance by ensuring the user control over the alarm system.
5	R-5	Lock door	Example: The user can lock the car doors with single press of button from the remote key to ensure the security of vehicle. Justification: Essential for guarding against unwanted entry and securing passengers and possessions inside the car.
6	R-6	Sound activate	Example: When the car is locked, audible confirmation tones are released, giving the user peace of mind that the doors are locked firmly. Justification: Gives prompt feedback on system operations, like locking or arming the alarm, which boosts user confidence.
7	R-7	Sound deactivate	Example: Enables users to manually turn off audio alerts or alarms to quiet the surroundings. Justification: Gives the user the option to decide when to turn on or off audible alerts, ensuring their comfort and convenience.

After extracting requirements through SA using text mining, we extracted the classification of requirements through which we identify the user-based and system-based requirements; those requirements that are system dependent and their behavior is dependent on the system are listed as system-based requirements, whereas those requirements which are user dependent and they are related to users are listed as user-based requirements as listed in Table 5. Their current ranking extracted (after requirement elicitation stored in the repository), prior ranking (attained through CBR) and new ranking (based on higher value of either from present ranking or prior ranking) are mentioned in Table 5. The requirement prediction after CBR and text mining by using the RPTSP model is shown in Table 5. The missing requirement ranking must be acquired to handle inconsistency

effectively, reducing complexity and ambiguities. The observation reveals that both approaches have different outcomes; rankings of R-2 and R-7 might have been overlooked by stakeholders or missing. Therefore, CBR is used to detect missing rankings with the prior ranking rather than assuming ranking requirements in development is creating incompleteness and ambiguity.

Table 5: Extracted ranking

Sr. #	Requirements ID	Classification of requirements	Identification of stakeholders	Present ranking	Prior ranking	New ranking	Prediction
1	R-1	User-based	Experience	5	4	5	R-3
2	R-2	System based	Non-Experience	–	4	4	R-1
3	R-3	System based	Experience	0	6	6	R-5
4	R-4	System based	Experience	1	3.5	3.5	R-2
5	R-5	User-based	Experience	4	4.5	4.5	R-4
6	R-6	System based	Experience	2	1	2	R-7
7	R-7	System based	Experience	–	3	3	R-6

4.1 Compare and Contrast PM with Existing Methodologies

The above-mentioned requirements are executed without PM (WPM), AHP and clustering, to relate the outcomes of sequence takeout from PM and WPM. Hence, Table 6 defines the implemented order of requirements created on priority through PM and WPM. In WPM methods, created order through an average calculation of these methods and acquired ordinary order for differentiation. As a result, the requirements ranking is changed after applying on PM and WPM methods mentioned in Table 6. Therefore, the sequence of requirements using PM is defined in column two, and the sequence of requirements using WPM in column three. In the PM, the R-3 score was the highest and the R-6 score the lowest, whereas in WPM, R-1 got the highest ranking, and R-2 got the lowest ranking. Afterwards, applying the requirements through both methods, results revealed that the PM members have more accurate requirement priority and satisfaction level than members WPM.

Table 6: Comparison of ranking

Sr. #	PM	WPM
1	R-3	R-1
2	R-1	R-5
3	R-5	R-4
4	R-2	R-7
5	R-4	R-3
6	R-7	R-6
7	R-6	R-2

We selected the test cases (see Table A3) based on requirements in order to validate the requirements that are more likely to contain errors than not. Following TCS, we rank test cases in order of importance to minimize execution time using two criteria: code coverage and test cases that are

frequently used. To determine the maximum error at an earlier stage, we sort them. Subsequently, all of these procedures assessed each participant's degree of satisfaction with their adoption of PM and WPM. Figs. 3–5 describe the evaluation of parameters on different satisfaction levels (see Table A1) of participants who contributed to the experiment and shows the results of EG and CG participants. Most participants were satisfied with RPTSP compared to those who are not employing it. The participant's satisfaction level on the y-axis and x-axis represents factorial analysis, respectively. The satisfaction level is achieved by more than 50 percent in both projects with maximum participants. For evaluation, we have used different rating scales, which are Satisfied (S), Highly Satisfied (HS), Neutral (N), Highly Dissatisfied (HD), and Dissatisfied (D). The participants of EG who used RPTSP were HS as compared to CG participants who did not use RPTSP. Whereas, the results indicate that RPTSP achieves better quality, understands customers' needs and produces good results compared to RPTSP.

To answer EQ-2, we implemented three selected cases in which CG participants used both AHP and clustering and their results shown in Table 7 in the same experiment while the EG used RPTSP and the results analyzed or compared through statistical analysis. These approaches are selected because they are the most frequently used existing approaches. Table 7 presents the level of satisfaction for all the participants after implementing RPTSP, clustering, and AHP techniques.

Table 7: Satisfaction level

Participants	RPTSP	AHP	Clustering
PMO	85	52	51
Sr	66	50	52
RA	88	44	42
TL	67	55	52
QA	72	58	57
Ds	71	58	55

We conducted paired sample tests on various approaches, including RPTSP, AHP, and clustering, to better understand the consequences and significance of each. For all groups, the results show different means, i.e., 0.724 of RPTSP, whereas clustering and AHP methods show means of 0.362 and 0.475, respectively. This indicates that the values of RPTSP scatter less from their mean value and are more reliable than other clustering and AHP methods.

The test values are different, proving that the experiment's performance is reliable and unbiased, lacking ambiguity. According to the obtained results, the mean value of the RPTSP is more in both datasets (dataset consists of the result extracted or generated after the experiment based on the participants feedback due to organization policy, we are not showing the complete information about the dataset but some evaluation parameters on different scales shown in Table A1 and demographic information shown in Table A2). The RPTSP dataset has a mean value of 0.718 and 0.734, while the dataset without RPTSP has mean values of 0.346 and 0.380. Therefore, our RPTSP model improves the prediction of requirements, selection, and prioritization of test cases. As FDR increases due to correct prioritization based on frequently and not frequently reuse test cases criteria by prioritizing test cases according to their reusability frequency. Thus, it proves that FDR positively depends on the correct TCP criteria. If the TCP criteria are not correct then FDR decreases as depicted from CG results while EG results depict the positive relation.

To answer EQ-3, we used accuracy (Accu) metrics and F measures [26]. SA were used for the correct selection of requirements. F measure is calculated through the combination of both precision (P) and recall (R), showing the overall efficiency of the optimum TCS process. F measure is calculated using Eq. (1).

$$F\ measures = \frac{2 * P * R}{P + R} \tag{1}$$

P is the relation of correctly specified and prioritized selected requirements to the total number of requirements accessible. R is the percentage of requirements correctly specified and prioritized to the total number of requirements accessible for reusable. $Accu$ is the relation of correctly classified requirements to the total number of requirements. $Accu$ is calculated using Eq. (2).

$$Accu = \frac{TP + TN}{FP + FN + TP + TN} \tag{2}$$

TN presents a true negative (similar requirements acquired for reusability), and FP represents a false positive (number of requirements but not selected). FN specifies false negatives (requirements retrieved for reusability but selected), while TP specifies true positives (requirements acquired for reusability and selected). The EQs results are shown in Fig. 6; the value in percentage is presented on the x-axis, whereas the metric name is presented on the y-axis.

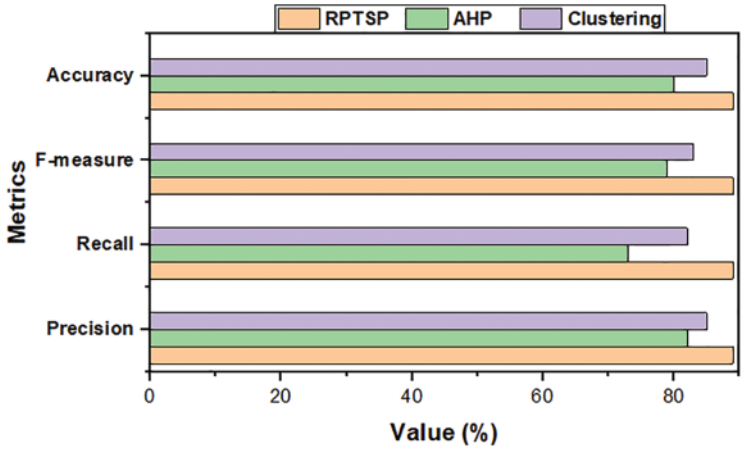

Figure 6: Value of metrics

To answer EQ-4, we used $APFD$ for predicted requirements which are mapped into test cases and calculated the frequency of most frequent failure test cases through code coverage criteria and generate parameters to calculate the priority and order of the test case. The different test cases are selected through the predicted requirement, and FDR was calculated using $APFD$ in the RPTSP model. $APFD$ is used to calculate the FDR over the prioritized test cases. The greater the value of FDR, the greater the number of defects that can be detected during TCS and TCP. $APFD$ is calculated using Eq. (3).

$$APFD = 1 - \frac{(TF_1 + TF_2 + TF_3 + \ldots TF_m)}{(mn)} + \frac{1}{2n} \tag{3}$$

In the above equation, TF_i represents the number of executed test cases, m identifies the number of faults, and n represents the number of test cases. The faults identified in each of the three cases are

depicted in Fig. 7. The test cases are presented on the *x*-axis, and FDR is presented on the *y*-axis. The results mentioned that FDR in all three cases is above 80% after the initial execution of test cases, whereas WPM shows the performance of FDR is above 20% in all three cases. RPTSP has higher APFD than WPM in all cases, which means that RPTSP criteria of prioritization is a key component and can identify higher FDR than WPM. RPTSP technique identifies maximum FDR in all cases, improving the performance, time, and cost.

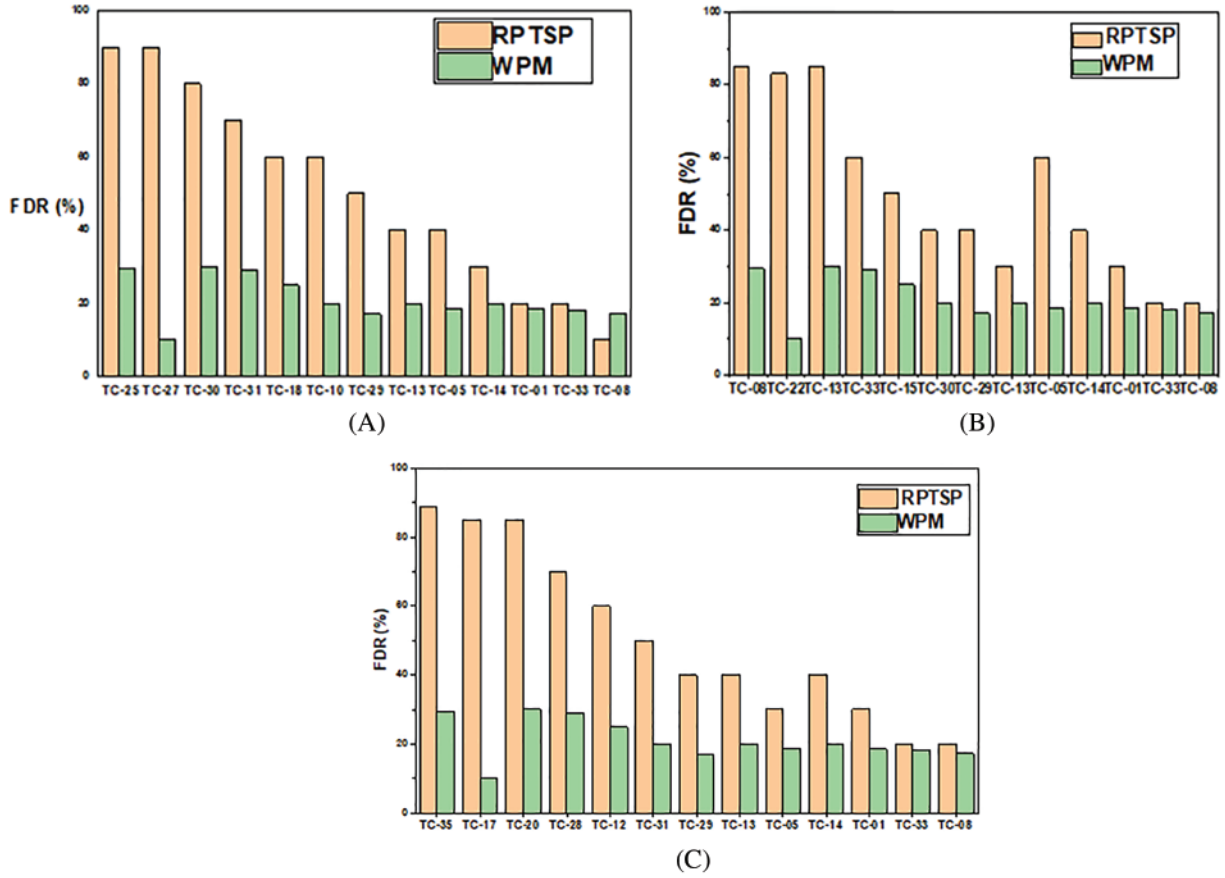

Figure 7: All three selected cases results (A, B and C)

Fig. 8 shows all projects' overall RPTSP model review and study findings. Participants' satisfaction levels are shown on the *y*-axis, and a parameter analysis (depicted in Table 3) is on the *x*-axis. The majority of participants in all projects reported level of satisfaction above 50%, which shows that the RPTSP model outperforms WPM.

Each project development strategy is labeled on the *x*-axis, and each participant's level of satisfaction is shown on the *y*-axis in Fig. 9. The results of all three cases using RPTSP and WPM; most participants are highly satisfied with the PM are modeled in Fig. 9. Individuals in case A who used our model reported higher levels of satisfaction and improved outcomes compared to participants who executed case A using WPM. The participants in cases B and C are similarly affected though. Cases B and C participants improved their quality and competency using the RPTSP model. According to the comparative analysis, the RPTSP model consistently outperformed alternative methods in

terms of customer satisfaction. The overall findings demonstrate that the RPTSP model outperforms conventional methods in terms of quality and customer needs satisfaction.

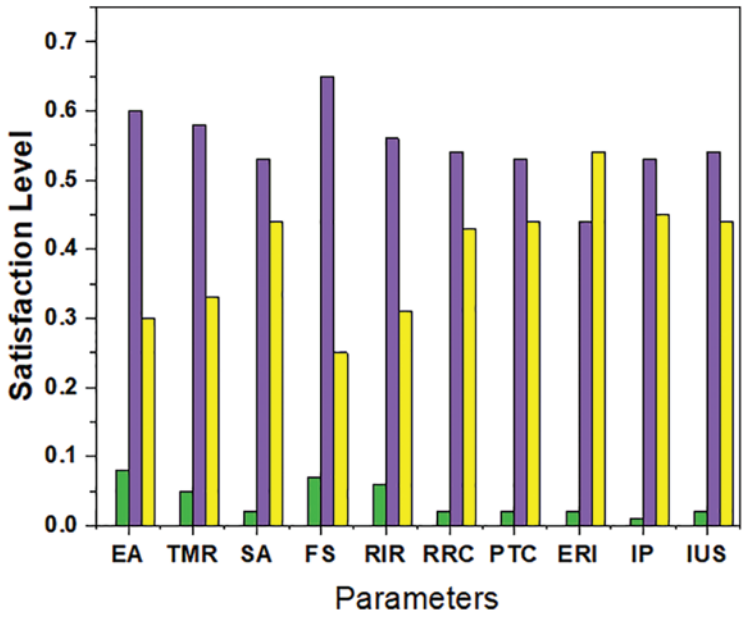

Figure 8: Review analysis

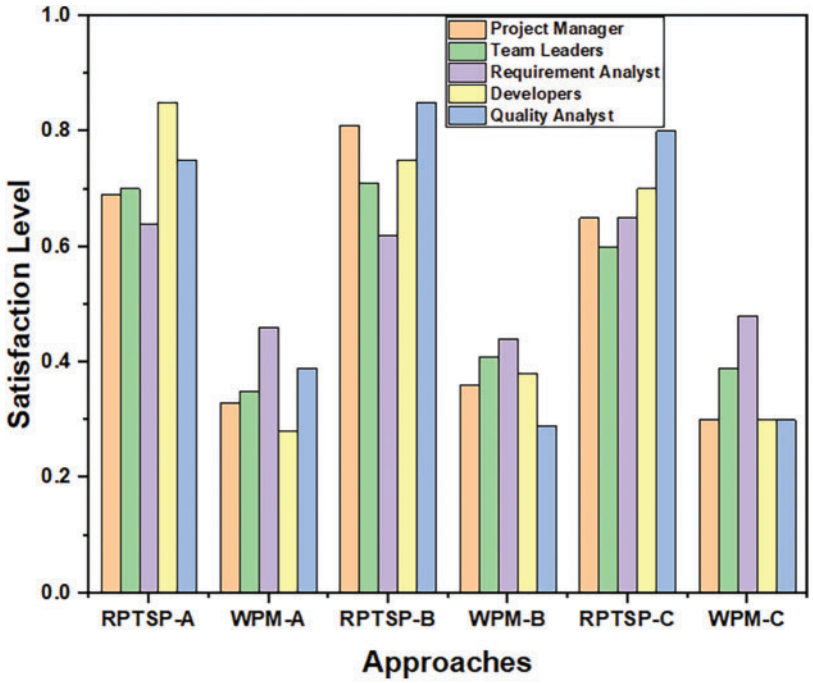

Figure 9: Overall models result

The answers of RQ1, RQ2, RQ3 and RQ4 conclude based on our research findings as following:

RQ1: The introduction section, we conclude that the requirements do have an effect on TCP, as depicted in Table 1 based on given parameters. These parameters are highlighted in different existing studies which we consider in the formulation of RPTSP and in the evaluation of RPTSP to improve the TCS and TCP process after change in requirements to reduce irrelevancy and similarity.

To answer the RQs, we used parameters mentioned in Table 3 to find the accuracy of reuse and selection of requirements (RQ2), evaluate the change impact analysis (RQ3) and identify the FDR (RQ4), respectively.

RQ2: The TCP process in the RPTSP model is greatly impacted by the appropriate reuse and requirement selection. The model uses a hybrid methodology that combines TCS and RP. The model seeks to enhance TCP through efficient reuse and requirement selection. More precisely, the model prioritizes requirements based on input from stakeholders and historical experiences by using CBR techniques to find comparable and relevant requirements. This technique increases the overall TCP procedure efficiency by ensuring that test cases correspond to the most important and frequently changing requirements.

RQ3: Yes, the change effect analysis that is part of the RPTSP model enhances the TCP process. The RP and clustering examine how requirements change over time. The model facilitates the identification of relevance, reuse opportunities, and modifications through the consideration of requirements similarity and its impact. The TCP procedure is guaranteed to adjust to changing software requirements thanks to this change effect analysis. As a result, the TCP process is more effective because of the model's capacity to anticipate changes and prioritize test cases appropriately.

RQ4: Indeed, increasing the FDR is one of the goals of the RPTSP model. A process of prioritizing test cases according to the expected and clustered requirements is integrated into the model. The RPTSP model makes sure the most significant and relevant test cases are run early in the testing process by utilizing the APFD metric. The findings show that in all three cases, the FDR attained using the RPTSP model is higher than 90%. This indicates that the model's prioritization mechanism helps find errors early on, which raises the process's overall FDR during software testing.

5 Practical Implications

The RPTSP model aims to improve the software testing process in agile environment by aligning requirements with test cases. The practical implications of this model are significant, offering improvements in accuracy, efficiency, and overall product quality. Scenario-based modeling helps in identifying primary requirements and user expectations. This leads to a clear specification of requirements, by reducing ambiguities. The categorization of requirements into user-based and system-based through SA, the model ensures that all stakeholder priorities are considered and conflicts are minimized. Using CBR to cluster requirements based on similarity helps in reusing previous knowledge and ranking requirements effectively. The use of semi-supervised K-means clustering method helps in organizing large amounts of data into meaningful clusters, reducing ambiguity and complexity. By applying the apriori algorithm, the model identifies frequent patterns in requirements, ensuring that test cases are selected based on relevance and historical usage. The APFD metric is used to prioritize test cases, ensuring that the most critical ones are executed first, leading to higher FDR.

The practical implications that demonstrate its applicability and benefits in real-world. Agile projects, for which we consider case B healthcare system. A healthcare is developing a system to manage records of patients, schedule appointments, and handle billing. Scenario-based modeling

outlined use cases such as "Patient record management" and "Appointment scheduling". Requirements were classified into user-based (e.g., appointment scheduling) and system-based (e.g., database management) using SA. CBR and semi-supervised K-means clustering organized requirements, ensuring that similar needs were grouped and prioritized effectively. The apriori algorithm highlighted frequently used requirements, and the APFD metric prioritized test cases for critical functionalities like patient data security. The requirement classification and clustering reduced ambiguities, leading to more accurate requirement predictions. Prioritized test cases based on historical usage and criticality ensured that the most important functionalities were tested first. Efficient clustering and prediction minimized redundant and irrelevant requirements, streamlining the testing process.

The RPTSP model offers a structured approach to aligning requirements and test cases, leading to significant improvements in the software testing process within agile environments. By focusing on requirement specification, clustering, prediction, and prioritized testing, the model enhances efficiency, accuracy, and overall product quality.

6 Conclusions

In this paper, we proposed the RPTSP model, which improves the redundant and irrelevant requirement selection process and FDR. The identification of parameters improves the requirement prediction and testing in ABE. In this study, we have identified the parameters that help identify redundant and irrelevant requirements to reduce the rate of fault detection during validation. Three different cases have been used to evaluate the RPTSP model. This research will help practitioners in the industry to improve the performance of agile-based software requirements and testing.

The evaluation shows the following outcomes:

- The RPTSP model extensively improved the redundancy and irrelevancy of requirements (more than 90%) at the initial level compared to other clustering and AHP methods.
- The evaluation metrics results show that the RPTSP model significantly outperforms other existing techniques to avoid irrelevant and redundant requirements.
- The result also described that FDR is improved through the predicted requirements, improving the irrelevancy in TCS and reducing ambiguity and complexity.
- Identification of relevant and redundant requirements demonstrates accuracy close to 100%, precision over 90%, and recall over 90%. As a result, our APFD rises as a result of relevant and non-redundant requirements, correct TCP and relevant TCS. The results demonstrate that, there is over 80% APFD at an earlier stage as compared to WPM (such as clustering and AHP).

7 Potential Limitations and Future Work

In this research, we align the testing process for new and change requirements by predicting accurate and redundant requirements against TCS. Hence, it improves the requirement validation process, we analyze that there is need to manage or more focus on requirement pre changes implementation, during change implementation and post change implementation as a future work. We can also automate the process by providing one platform for aligning the requirements and testing activities. There is need to enhance RPTSP model in terms of both static and dynamic configuration testing processes due to modern tools and techniques. In the future, we are planning to implement this model through a proper tool that will solve the redundancy and irrelevancy of requirements and identify more faults based on the selection and prediction of requirements. In the future, we are trying to plan more RQs to evaluate FDR and compare it with more existing strategies.

Acknowledgement: Not applicable.

Funding Statement: Not applicable.

Author Contributions: Conceptualization, Irum Ilyas and Yaser Hafeez; Formal analysis, Irum Ilyas, Sadia Ali, Mamoona Humayun and Muhammad Aqib; Funding acquisition, Nabil Almashfi and Ghadah Alwakid; Project administration, Mamoona Humayun, Nabil Almashfi and Ghadah Alwakid; Supervision, Yaser Hafeez and Mamoona Humayun; Validation, Irum Ilyas, Sadia Ali and Mamoona Humayun; Writing—original draft, Irum Ilyas; Writing—review & editing, Irum Ilyas and Muhammad Aqib. All authors reviewed the results and approved the final version of the manuscript.

Availability of Data and Materials: Due to privacy considerations, the data supporting the study's conclusions cannot be shared.

Ethics Approval: Not applicable.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] M. Q. Riaz, Fateh-ur-Rehman, B. Maqbool, and W. H. Butt, "Customization of requirement engineering best practices for Pakistan software industry," in *2018 Int. Conf. Comput. Math. Eng. Technol. (iCoMET)*, Sukkur, Pakistan, IEEE, Mar. 2018, pp. 1–6. doi: 10.1109/ICOMET.2018.8346347.

[2] S. -W. Lee and T. Nakatani, "Requirements engineering toward sustainable world," in *Communications in Computer and Information Science*, Singapore: Springer Singapore, 2016, vol. 671. doi: 10.1007/978-981-10-3256-1.

[3] S. Ali, Y. Hafeez, S. Hussain, and S. Yang, "Enhanced regression testing technique for agile software development and continuous integration strategies," *Softw. Qual. J.*, vol. 28, no. 2, pp. 397–423, Jun. 2020. doi: 10.1007/s11219-019-09463-4.

[4] R. D. Estrada-Esponda, M. López-Benítez, G. Matturro, and J. C. Osorio-Gómez, "Selection of software agile practices using analytic hierarchy process," *Heliyon*, vol. 10, no. 1, Jan. 2024, Art. no. e22948. doi: 10.1016/j.heliyon.2023.e22948.

[5] R. Wieringa, "Requirements engineering since the year one thousand," in *2017 IEEE 25th Int. Requirements Eng. Conf. (RE)*, Lisbon, Portugal, IEEE, Sep. 2017, pp. 480–481. doi: 10.1109/RE.2017.25.

[6] Y. Lu *et al.*, "How does regression test prioritization perform in real-world software evolution?," in *Proc. 38th Int. Conf. Softw. Eng.*, Austin, TX, USA, ACM, May 2016, pp. 535–546. doi: 10.1145/2884781.2884874.

[7] S. Ali, Y. Hafeez, S. Hussain, S. Yang, and M. Jamal, "Requirement prioritization framework using case-based reasoning: A mining-based approach," *Expert Syst.*, vol. 38, no. 8, Dec. 2021, Art. no. e12770. doi: 10.1111/exsy.12770.

[8] S. Ali, M. Imran, Y. Hafeez, T. R. Abbasi, W. Haider and A. Salam, "Improving component based software integration testing using data mining technique," in *2018 12th Int. Conf. Math. Actuar. Sci., Comput. Sci. Stat. (MACS)*, Karachi, Pakistan, Nov. 2018, pp. 1–6. doi: 10.1109/MACS.2018.8628368.

[9] M. Shameem, R. R. Kumar, M. Nadeem, and A. A. Khan, "Taxonomical classification of barriers for scaling agile methods in global software development environment using fuzzy analytic hierarchy process," *Appl. Soft Comput.*, vol. 90, no. 6, May 2020, Art. no. 106122. doi: 10.1016/j.asoc.2020.106122.

[10] B. Şahin, "Route prioritization by using fuzzy analytic hierarchy process extended dijkstra algorithm," *J. ETA Marit. Sci.*, vol. 7, no. 1, pp. 3–15, 2019. doi: 10.5505/jems.2019.39306.

[11] N. Mahmood *et al.*, "Mining software repository for cleaning bugs using data mining technique," *Comput. Mater. Contin.*, vol. 69, no. 1, pp. 873–893, 2021. doi: 10.32604/cmc.2021.016614.
[12] S. Mondal and R. Nasre, "Colosseum: Regression test prioritization by delta displacement in test coverage," *IEEE Trans. Softw. Eng.*, vol. 48, no. 10, pp. 4060–4073, Oct. 2022. doi: 10.1109/TSE.2021.3111169.
[13] R. Yan, Y. Chen, H. Gao, and J. Yan, "Test case prioritization with neuron valuation based pattern," *Sci. Comput. Program.*, vol. 215, Mar. 2022, Art. no. 102761. doi: 10.1016/j.scico.2021.102761.
[14] A. Samad, H. Mahdin, R. Kazmi, and R. Ibrahim, "Regression test case prioritization: A systematic literature review," *Int. J. Adv. Comput. Sci. Appl.*, vol. 12, 2021. doi: 10.14569/issn.2156-5570.
[15] A. Gupta, N. Mishra, A. Tripathi, M. Vardhan, and D. S. Kushwaha, "An improved history-based test prioritization technique technique using code coverage," in *Adv. Comput. Commun. Eng. Technol.: Proc. 1st Int. Conf. Commun. Comput. Eng.*, Rajshahi, Bangladesh, Springer International Publishing, 2015.
[16] X. Wang and S. Zhang, "Cluster-based adaptive test case prioritization," *Inf. Softw. Technol.*, vol. 165, no. 2, Jan. 2024, Art. no. 107339. doi: 10.1016/j.infsof.2023.107339.
[17] A. M. Pitangueira, R. S. P. Maciel, and M. Barros, "Software requirements selection and prioritization using SBSE approaches: A systematic review and mapping of the literature," *J. Syst. Softw.*, vol. 103, no. 14, pp. 267–280, May 2015. doi: 10.1016/j.jss.2014.09.038.
[18] M. A. Siddique, W. M. N. Wan-Kadir, J. Ahmad, and N. Ibrahim, "Hybrid framework to exclude similar and faulty test cases in regression testing," *Baghdad Sci. J.*, vol. 21, no. 2, Feb. 2024, Art. no. 0802. doi: 10.21123/bsj.2024.9710.
[19] A. Bajaj and O. P. Sangwan, "A systematic literature review of test case prioritization using genetic algorithms," *IEEE Access*, vol. 7, pp. 126355–126375, 2019. doi: 10.1109/ACCESS.2019.2938260.
[20] H. Mirzaei and M. R. Keyvanpour, "Reinforcement learning reward function for test case prioritization in continuous integration," in *2022 9th Iran. Joint Congress Fuzzy Intell. Syst. (CFIS)*, Bam, Iran, Islamic Republic of IEEE, Mar. 2022, pp. 1–6. doi: 10.1109/CFIS54774.2022.9756464.
[21] M. Khatibsyarbini *et al.*, "Trend application of machine learning in test case prioritization: A review on techniques," *IEEE Access*, vol. 9, pp. 166262–166282, 2021. doi: 10.1109/ACCESS.2021.3135508.
[22] A. Ali *et al.*, "A data mining technique to improve configuration prioritization framework for component-based systems: An empirical study," *Inf. Technol. Control*, vol. 50, no. 3, pp. 424–442, Sep. 2021. doi: 10.5755/j01.itc.50.3.27622.
[23] X. Wang and H. Zeng, "History-based dynamic test case prioritization for requirement properties in regression testing," in *Proc. Int. Workshop Cont. Softw. Evol. Deliv.*, Austin, TX, USA, ACM, May 2016, pp. 41–47. doi: 10.1145/2896941.2896949.
[24] J. C. S. Coutinho, W. L. Andrade, and P. D. L. Machado, "Requirements engineering and software testing in agile methodologies: A systematic mapping," in *Proc. XXXIII Brazil. Symp. Softw. Eng.*, Salvador, Brazil, ACM, Sep. 2019, pp. 322–331. doi: 10.1145/3350768.3352584.
[25] N. L. Atukorala, K. C. Chang, and K. Oyama, "Situation-oriented evaluation and prioritization of requirements," in *Requirements Eng. Toward Sustain. World: Third Asia-Pacific Symp. APRES 2016*, Nagoya, Japan, Springer Singapore, Nov. 10–12, 2016, vol. 671, pp. 18–33.
[26] Y. Hafeez, S. Ali, N. Jhanjhi, M. Humayun, A. Nayyar and M. Masud, "Role of fuzzy approach towards fault detection for distributed components," *Comput. Mater. Contin.*, vol. 67, no. 2, pp. 1979–1996, 2021. doi: 10.32604/cmc.2021.014830.
[27] M. Shameem, A. A. Khan, M. G. Hasan, and M. A. Akbar, "Analytic hierarchy process based prioritisation and taxonomy of success factors for scaling agile methods in global software development," *IET Softw.*, vol. 14, no. 4, pp. 389–401, Aug. 2020. doi: 10.1049/iet-sen.2019.0196.
[28] R. Santos, A. Albuquerque, and P. R. Pinheiro, "Towards the applied hybrid model in requirements prioritization," *Procedia Comput. Sci.*, vol. 91, pp. 909–918, 2016. doi: 10.1016/j.procs.2016.07.109.
[29] S. Dhingra, G. Savithri, M. Madan, and R. Manjula, "Selection of prioritization technique for software requirement using fuzzy logic and decision tree," in *2016 Online Int. Conf. Green Eng. Technol. (IC-GET)*, Coimbatore, India, IEEE, Nov. 2016, pp. 1–11. doi: 10.1109/GET.2016.7916822.

[30] R. Naseem *et al.*, "Empirical assessment of machine learning techniques for software requirements risk prediction," *Electronics*, vol. 10, no. 2, pp. 168, Jan. 2021. doi: 10.3390/electronics10020168.
[31] F. Shao, R. Peng, H. Lai, and B. Wang, "DRank: A semi-automated requirements prioritization method based on preferences and dependencies," *J. Syst. Softw.*, vol. 126, no. 6, pp. 141–156, Apr. 2017. doi: 10.1016/j.jss.2016.09.043.
[32] P. Achimugu and A. Selamat, "A hybridized approach for prioritizing software requirements based on K-means and evolutionary algorithms," *Comput. Intell. Appl. Model. Control*, pp. 73–93, 2015.
[33] X. Wang and H. Zeng, "Dynamic test case prioritization based on multi-objective," in *15th IEEE/ACIS Int. Conf. Softw. Eng. Artifi. Intell. Network Parallel/Distrib. Comput. (SNPD)*, Las Vegas, NV, USA, IEEE, Jun. 2014, pp. 1–6. doi: 10.1109/SNPD.2014.6888744.
[34] Z. Q. Zhou, C. Liu, T. Y. Chen, T. H. Tse, and W. Susilo, "Beating random test case prioritization," *IEEE Trans. Reliab.*, vol. 70, no. 2, pp. 654–675, Jun. 2021. doi: 10.1109/TR.2020.2979815.

Appendix A

Table A1: Evaluation of parameters on different satisfaction level

Parameters	HS		S		N		DS		HD	
	EG	CG	EG	CG	EG	CG	EG	CG	EG	CG
RIR	24	1	65	25	6	26	2	24	0	66
EA	43	1	52	39	2	42	2	43	0	54
TMR	35	1	55	38	7	36	2	35	0	57
SA	42	2	53	41	2	42	1	42	0	55
IP	43	2	52	44	2	41	1	43	0	52
FS	53	2	43	38	2	51	1	54	0	43
HIR	44	2	53	43	1	40	1	44	0	51
RPP	42	2	55	44	2	44	1	43	0	52
PDR	43	2	54	45	2	41	1	41	0	54
ERI	32	3	53	42	5	33	3	30	0	57
PTC	33	3	59	33	5	30	3	33	0	59
RCR	53	2	58	38	2	50	1	54	0	43
RRR	44	2	42	38	1	40	1	45	0	51
PAI	42	2	53	43	2	44	1	43	0	52
ICA	43	2	54	44	2	41	1	44	0	52
IUS	29	2	60	32	8	32	2	31	0	60
FDR	24	1	65	25	6	26	2	24	0	66
FCTC	35	1	55	38	7	36	2	35	0	57

Table A2: Demographic information of participants

Participants	Experience	CG	EG
Project manager owner	</> 2 Years	2	2
Team leaders	</> 2 Years	2	2
Requirement analyst	</> 1 Years	2	2
Developers	</> 2 Years	2	2
Stakeholders/Customers	</> 4 Years	2	2
Quality analyst	</> 2 Years	2	2

Table A3: Detailed test cases and their outcome

Test case id	Test scenario	Test case	Pre-condition	Test steps	Expected result	Actual result	Status
TC-001	Unlock car door with valid key fob signal	Verify that the car door unlocks when the key fob sends a valid unlock signal	1. The car is locked 2.The key fob is within the range of car receiver	1. Press the unlock button on key fob 2. Observe the car door locks	Car door unlocks and the lights flash	Car door unlock	Pass
TC-002	Unlock car door with manual key	Verify that the car door unlocks when the manual key is used	1. The car is locked 2. The manual key is available	1. Insert the manual key into car door lock 2. Move the key to unlock position 3. Observe the car door locks	Car door unlocks and door can be opened	Car door unlock	Pass
TC-003	Unlock car door with smart entry system	Verify that the car door unlocks when smart entry system detects the key fob in proximity	1. The car is locked 2. The key fob is within the proximity range of car smart entry system	1. Approach car with key fob in your possession 2. Attempt to open the car door by pulling handle	Car door unlocks and door opens without pressing any button on key fob	Car door unlock	Pass
TC-004	Attempt to unlock car door with an invalid key fob signal	Verify that the car door does not unlock when an invalid key fob signal is sent	1. The car is locked 2. An incorrect key fob is available	1. Press the unlock button on the invalid key fob 2. Observe the car door locks	The car door remains locked and no unlocking occurs	Car door locked	Pass
TC-005	Unlock car door with low battery in key fob	Verify the car door unlocks when the key fob has a low battery	1. The car is locked 2. The key fob has a low battery but is still operational	1. Press the unlock button on the key fob with a low battery 2. Observe car door locks	The car door unlocks but there may be a delay or reduced signal strength	Car door unlock but a delay	Pass