ISSN 2409-2665
Journal of Logistics, Informatics and Service Science
Vol. 11 (2024) No. 10, pp. 65-85
DOI:10.33168/JLISS.2024.1005

A Comprehensive Assessment Model for Requirement-based Test Case Prioritization: Integrating Internal Factors and Dependency Analysis through Expert Evaluation

Ani Rahmani [1,2], Sabrina Ahmad [2], Intan Ermahani A Jalil [2], Seok-Won Lee [3,4]

[1] Dept. of Computer Engineering and Informatics, Bandung State Poltytechnic, Indonesia

[2]Faculty of Information and Communication Technology, Universiti Teknikal Malaysia Melaka, Malaysia

[3] Dept. of Software and Computer Engineering, Ajou University, Suwon, South Korea

[4] Dept. of Artificial Intelligence, Ajou University, Suwon, South Korea

anirahma@polban.ac.id

Abstract. Test case prioritization (TCP) is a critical component of regression testing in agile software development. Requirement-based TCP leverages information from software requirements to improve testing effectiveness, but gathering and assessing relevant requirement factors remains a challenge. This paper proposes a comprehensive assessment model for requirement-based TCP that incorporates both internal factors (complexity, change impact, and prioritization) and requirement dependencies. The model employs pairwise comparison (PC) and expert evaluation techniques to assign weights to the factors. The assessment process is validated using the iTrust medical record system as a case study, with four experts from diverse software engineering backgrounds participating in the evaluation. The resulting weighted factors provide a quantitative basis for designing more effective TCP strategies. The proposed model contributes to the advancement of requirement-based testing and offers practical insights for software practitioners seeking to optimize their TCP processes.

Keywords: regression testing, test case prioritization, requirement-based testing, requirement-based TCP, pairwise comparison

1. Introduction

Test case prioritization (TCP) is a well-known technique in regression testing designed to improve testing effectiveness at the execution level. One of the main challenges in regression testing is the limited time available (Yaraghi et al., 2023). Therefore, it is required to reduce the execution time in the testing process. It is particularly relevant in agile software development (ASD), due to sprints are typically short and require a fast-testing process. Preliminary research shows a significant advantage of TCP: if the testing process is interrupted, it likely means that errors or faults have already been detected. In other words, existing faults can be found without executing all test cases.

TCP performs systematically scheduling all test cases (test suite) to decide the test case priorities to be executed earlier based on determinant criteria (Prado Lima & Vergilio, 2020; Silvarajoo & Hazim Alkawaz, 2022; Yaraghi et al., 2023). The main aim of TCP is to ensure that testing execution is more efficient. It is possible to execute because TCP prioritizes the highest potential test case in terms of fault detection. As a result, in the specific time, the number of faults detected using TCP tends to be higher compared to the conventional way of executing the test suite sequentially.

There are two important factors in the TCP development, which are the TCP method used and the appropriate technique or algorithm to implement TCP (Hemmati, 2019). The method serves as an input for the TCP implementation technique. There are several TCP approaches, such as Risk-based, Search-based, Fault-based, Model-based, Modification-based, Coverage-based, Similarity-based, Requirement-based, User Interface-based, History-based, Mutation-based, and Hybrid--combining more than one (Muhammad Khatibsyarbini et al., 2018; Rahmani et al., 2021)**.**

Requirement-based TCP is an approach that uses information or factors present in the software requirements as a basis for conducting test case prioritization (Hasnain et al., 2021; Muhammad Khatibsyarbini et al., 2018). Unlike other methods, the TCP practice based on requirements is limited. The software contains several requirements, and every requirement has its characteristics in various aspects. We refer the characteristics to it as internal factors from the requirement. On the other hand, every requirement will be interacted with other requirements. One kind of the requirement interaction is requirement dependencies. Therefore, it is significantly important to study the utilization of requirements factors to enhance the effectiveness of testing execution, particularly, the TCP study. In general, software requirements consist of functional and non-functional requirements. In this paper, the discussion is limited to functional requirements.

Several researchers have carried out requirements-based TCP studies. A Study by Rahmani et al. (Rahmani et al., 2021) reviewed at least 12 studies regarding requirements-based TCP. Two studies focus on requirements dependency; one uses requirements clustering, the other uses requirements coverage, and most studies utilize the requirements risk parameter. On the other hand, previous studies adopted many techniques or algorithms for prioritization. For example Modified-Ant Colony Optimization (M-ACO) technique (Silvarajoo & Hazim Alkawaz, 2022), Additional Greedy method Call sequence (Chi et al., 2020), Genetic Algorithms (Di Nucci et al., 2020; Habtemariam & Mohapatra, 2019; Mishra et al., 2019), Firefly Optimization Algorithm (M Khatibsyarbini et al., 2019), Multi-objective particle swarm optimization method (Samad et al., 2021), and utilization of Deep Learning (Sharif et al., 2021).

In the common studies of requirement-based TCP, generally, the studies focus on analyzing the effectiveness of TCP based on the techniques or algorithms used. Meanwhile, the information regarding relevant requirements factors is provided through assessment. The assessment objective can be biased and can stem from various factors such as assessor’s preferences and the competencies needed. In several studies, researchers in this field tend to conduct assessments by specifying the acceptable range of values to access each requirement factor, while excluding the value assigned. It has the potential to affect the assessment process highly subjective.

The main objective of this paper is to present a process model for assessing internal factors of requirements and dependencies between requirements for TCP development. The assessment result is the value weighted and calculated, then obtained quantitatively to prioritize test cases. The assessment model serves as a guideline for conducting assessments. Using the model, the assessment processes can be made objective and standardized, so that the assessment calculation result can guarantee that the prioritization process is more accurate.

This paper is organized following the structure: the introduction, section 2 focuses on the related works and literature review. Section 3 explains the research framework and methodology. Section 4 discusses the result of assessment model implementation, validated by the experts through trial. Finally, section 5 presents conclusion and future works.

2. Related Work and Literature Review

A crucial aspect of implementing requirements-based TCP is the assessment process of requirements factors. In this context, the factors are internal factors and dependencies between requirements. To support the assessment process, a literature review was conducted to identify key factors for the development of requirements-based TCP.

2.1. Internal Factors of Requirement

Internal factors of requirements are used in several requirements-based TCP studies. Some of these include utilization of requirements-risk (Hettiarachchi et al., 2014, 2016; Hettiarachchi & Do, 2019; Ma et al., 2016; Srikanth et al., 2016; Yoon et al., 2012). Krishnamoorthi's study (Krishnamoorthi & Sahaaya Arul Mary, 2009) focused on examining six requirement factors, three of which are associated with initial version and regression testing. The three factors for the initial version testing are customer priority, changes in requirements, and implementation complexity. For regression testing, the factors include usability, application flow, and fault impact. While (Srikanth et al., 2005) researched requirement-based TCP using four requirement factors. These include requirement volatility, customer priority, implementation complexity, and fault proneness of the requirements. The results obtained were known as the Prioritization of Requirements for Test (PORT) system. The internal factors, such as requirement complexity, change impact requirement, and requirement prioritization, were further elaborated on in the following section.

2.1.1 Requirement Complexity

In the software development, it is particular to consider both software and requirements complexity. Nonetheless, quoting (Cardoso, 2014) and (Apurva, 2021) stated that the high complexity requirements frequently in a higher occurrence of errors, defects, exceptions. It leads to elevated costs during the development, testing, and maintenance processes.

In ASD, a common practice among software engineers is to relate the complexity of project requirements to the effort required. They use a metric called story points to estimate this effort. The more complex a requirement, the greater the effort required, and consequently increase the probability encountering errors, faults, and other risks. The team typically applies user stories, considered as requirements to be implemented in their ASD sprints. These user stories are accompanied by story points, which play an important role in project planning and resource allocation. Several studies have been widely conducted on story points, including (Fernández-Diego et al., 2020; Kulasinghe, 2021; Salmanoglu et al., 2017; Tawosi et al., 2022). This research introduced automatic techniques for assessing story points, as manual assessment by experts is perceived as vulnerable, highly subjective, and potentially inconsistent. However, expert assessment of story points is widely used in practical software development.

The activity of determining of requirement complexity is carried out in the early stages. The value of user story (requirements) value is used to estimate the overall project. A survey conducted by (Kulasinghe, 2021), stated that 61.67% of the industry used story points to estimate complexity, often referring to Fibonacci numbers (Scott & Pfahl, 2018) in many cases. This sequence includes generating numbers by adding a series of preceding number.

2.1.2 Change Impact Requirement

During the software development process, changes to requirements are highly likely to occur and are often difficult to avoid. In a research related to ASD, (Saher, 2017) stated that a requirement change is a significant issue worthy of continuous discussion. Its management is perceived as a crucial challenge, as any mishandling could lead to project failure. Furthermore, a single requirement change can potentially impact other aspects of the project. The analysis of the impact of requirements changes has been a discussion subject among requirements engineering studies, such as the works of (Saher, 2017) and (Akbar et al., 2020). Some research suggested automating the analysis on the impact of a requirement change using various techniques. In contrast, manual methods adopted by experts remain prevalent due to their perceived precision.

2.1.3 Requirement Prioritization

In simple terms, requirement prioritization (RP) is a process used to determine high-priority requirements, although its significance goes beyond that. RP is a crucial stage in requirement engineering, aimed to establish the most appropriate implementation order based on needs. Hujainah (Hujainah et al., 2018) stated that it is generally in line with the interests of stakeholders. In more detail, Noviyanto (Noviyanto et al., 2023) stated that prioritization criteria typically take into account factors such as stakeholder preferences, functionality, cost, processing time, risk considerations, or a business perspective. Meanwhile, Hudaib et al. (Hudaib et al., 2018) reported that although RP can be seen from different perspectives, it is performed based on specific interests or its implementation.

Some researchers have explored RP techniques, using various approaches. For instance, a literature review by Hujainah et al. (Hujainah et al., 2018) identified 108 RP techniques from 122 research papers. Meanwhile, Hudaib et al. (Hudaib et al., 2018) compared RP techniques, and Borhan et al. (Borhan et al., 2019) identified those specifically used in ASD. Additionally, Hujainah et al. (Hujainah et al., 2018) categorized RP techniques into three types, namely manual, semi-automated, and fully automated.

2.2. Requirement Dependency

All requirements defined in the elicitation phase of the development process cannot be treated independently. These requirements are interconnected and influence each other in a complex manner (Dahlstedt & Persson, 2005). However, quoting (Carlshamrea et al., 2001) and (Deshpande et al., 2019), (Deshpande et al., 2020) stated that approximately 80% of the software requirements are interdependent. Neglecting the issue of dependency among requirements in the software development process could be potentially detrimental and may lead to failure.

Requirement dependency tends to influence various decisions and activities in software development (Dahlstedt & Persson, 2005). Furthermore, Li’s et al. study (Li et al., 2012) stated that it plays a crucial role in change propagation analysis in software, specifically at the requirement level. In prior research, a shared approach is evident in which requirement dependency is categorized into various types, forming a comprehensive model. The following research has made significant contributions to this model: (Dahlstedt & Persson, 2005; Li et al., 2012; Zhang et al., 2014). The use of requirement dependency for requirement-based TCP investigations was conducted by Abbas et al. (Abbas et al., 2019) and Vescan et al. (Vescan et al., 2017, 2021).

3. Methodology

A research framework designed for the purpose of this investigation, consisting of stages that are globally divided into several activities. It begins with the assessment modelling process, progresses to the identification of experts performing the assessment, to the determination of the Software Under Test (SUT) for the case research, and the assessment implementation trial by experts, as well as the calculation of requirement factor weight. The stages of these activities are shown in Fig.1.

3.1. Requirement Factors Determination

In designing the assessment model, the first step is to determine the first step is to determine the requirement parameters or factors to be assessed. The parameters used are further divided into two groups, namely internal factors, and requirements dependency. The rationale behind this categorization is to address not only the factors within each requirement (internal factors) but also to consider the interaction and relationships between requirements, specifically their dependencies. The two parameters are explained as follows.

3.1.1. Internal Factors

This research adopted three internal factors, namely requirement complexity (RCX), change impact requirement (RCG), and requirement prioritization (RPR). We determine these three factors because each requirement is strongly linked to them. Each requirement has a different complexity, which impacts the chances of errors occurring when implemented. The more complex a requirement, the higher the possibility of a fault or error occurring, and vice versa. Furthermore, in terms of the impact factor when a requirement changes. Requirements interact with each other. Some requirements have a significant impact when changed on the whole software, or other requirements. That impact will determine the possibility of error occurring during implementation. The third factor is requirements prioritization. This factor is essential to consider because, in software development, a requirement has a priority level for implementation compared to other requirements. This priority can be viewed from multiple perspectives. In this context, requirements with high priority must be ensured to be safer from faults due to their critical role in the system.

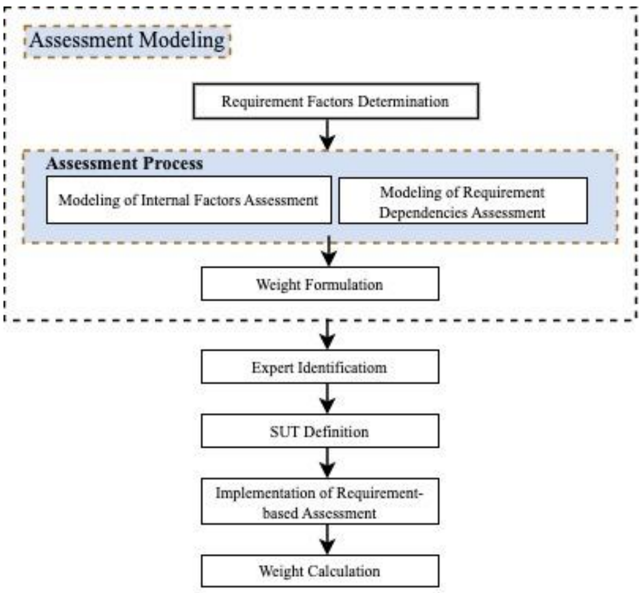

Assessment Modeling
Requirement Factors Determination
Assessment Process
Modeling of Internal Factors Assessment
Modeling of Requirement Dependencies Assessment
Weight Formulation
Expert Identificatiom
SUT Definition
Implementation of Requirement-based Assessment
Weight Calculation

Fig. 1: Research Framework

Before assessing these internal factors, experts conduct a PC assessment. PC is an age-old method that remains popular to this day to obtain preferences for each compared pair. This approach includes determining the significance of one criterion in relation to another concerning a specific goal (Oswaldo et al., 2014). To perform these comparisons effective of rely, a scale indicating how many times one element is more important or dominant than the other, with regard to the criterion under evaluation, is needed (Saaty, 2002). PC is measured using a linear scale proposed by Saaty in 1980 and is part of the Analytic Hierarchy Process (AHP). This assessment was aimed at comparing the relative importance of one factor to another, referring to the Saaty scale.

3.1.2. Requirement Dependency Type

The objective of the requirement dependency model is to categorize and describe the extent to which one requirement depends on another. To determine these dependency types, various methods are available. The types of dependency model used in this research were introduced by (Li et al., 2012), namely precondition/requires (PR), similar_to (SM), satisfies (ST), constraint (CO), and refines/refined to (RE).

3.2. Assessment Process

The assessment process is carried out to get values for each parameter, and experts perform this task. In this case, each expert, based on their expertise in the software engineering, assesses all the requirements in the software object using the designated parameters.

3.2.1 Assessment of Internal Factors of Requirement

The assessment of internal factors includes the evaluation of requirement complexity, change impact requirement, and requirement prioritization. However, before this process, experts assessed the pairwise levels for each internal factor. Table 1 shows the PC matrix for these three internal factors. The diagonal cells contain a value of 1, indicating dependency between identical factors. In this case, experts are only required to fill the cells in blue with the Pairwise Saaty levels (Saaty, 2002) while those in white would be automatically filled.

Table 1. Pairwise Comparison (PC) Assessment on Internal Factors

	RCX	**RCG**	**RPR**
RCX	1.00		
RCG		1.00	
RPR			1.00

After filling in the pairwise matrix is the priority vector calculation stage to get the final weight each factors. The stages in this process are: a) normalize the matrix by adding up all the values in the columns and dividing each element by the total of its corresponding column values, and b) calculate the criterion weight (CW) by averaging all the values in each row of the normalized matrix. CW serves as an approximation of the priority vector values. However, to ensure the acceptability of these values, the consistency ratio (CR) must be calculated to verify their consistency.

Next is calculating the CR value with the following steps:

1. Multiply the Saaty scale mapping by CW.
2. Determine the weighted sum value (WS) by adding up all rows of values in the product matrix of elements with the weight criteria.
3. Determine the maximum eigenvalue (λ_max) by averaging all the values and dividing the weighted sum value by the weight criteria value in each row (WS/CW), Formula 1. This λ_max value results from an approach whose value is always closer to the n value from the right. Saaty stated that the law λ_max≥n will always be fulfilled.

$$\lambda_{max} = \frac{WS/CW_1 + WS/CW_2 + \cdots .WS/CW_n}{n} \quad (1)$$

4. Calculate the consistency index (CI) using formula (2), where n is the number of criteria to be compared.

$$CI = \frac{\lambda_{max} - n}{n-1} \quad (2)$$

5. Calculate the CR by dividing the CI value by the random index (RI), as can be seen in formula 3, where the RI value is based on Table 2.

$$CR = \frac{CI}{RI} \quad (3)$$

Table 2. Random Index (RI) Standard Value

n	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
RI	0	0	0.58	0.9	1.12	1.24	1.32	1.41	1.45	1.49	1.51	1.48	1.56	1.57	1.59

This stage is repeated until the expert achieves a consistency ratio (CR) less than 0.1, which, according to Saaty (Saaty, 2002), indicates the correctness and consistency of the assessment process. Furthermore, the assessment process for the three internal factors is stated as follows:

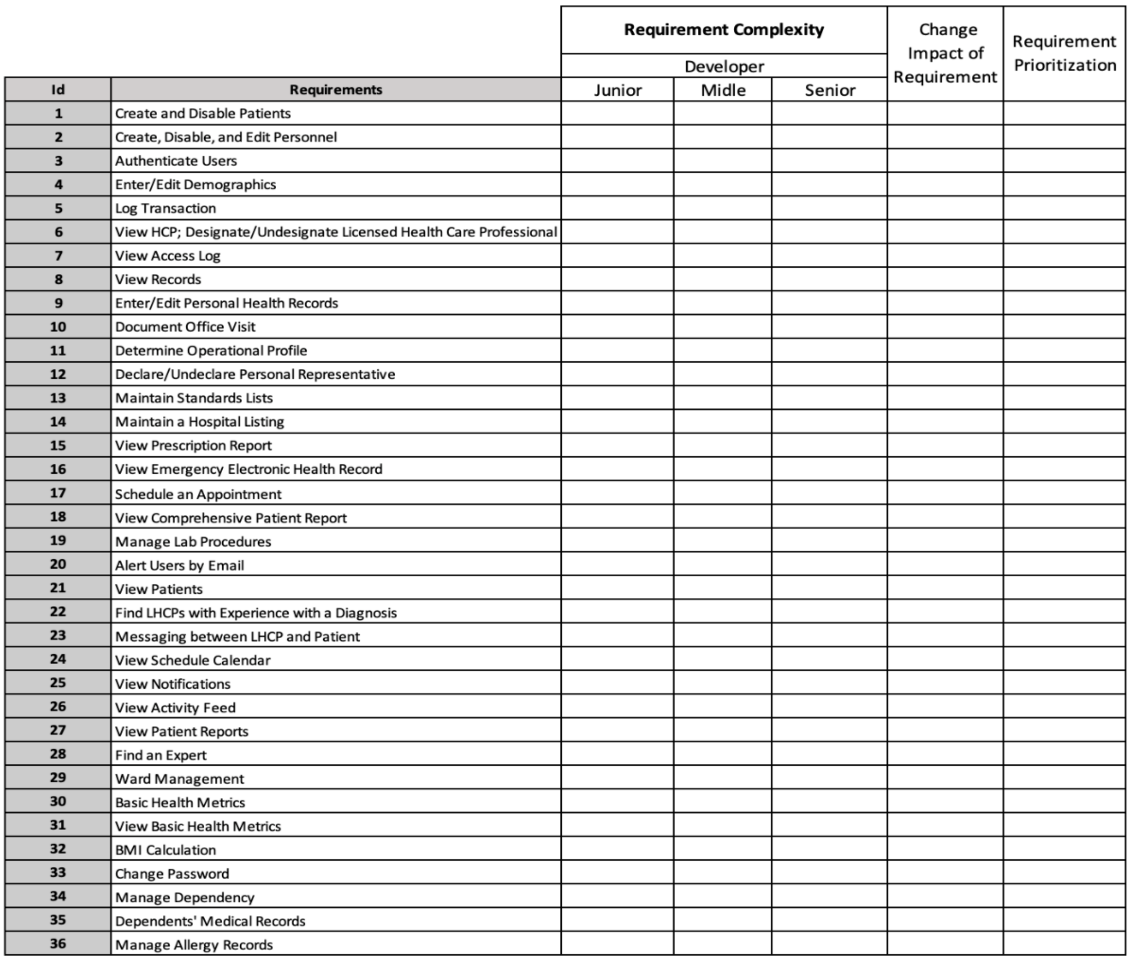
1. Assessment of Requirement Complexity
   An expert will assess requirement complexity by assigning a story point value. Referring to the explanation in section 2.1.1, requirement complexity is interpreted as the perception of an expert when the requirement is implemented by each of the three developer levels, namely junior, middle, and senior. To determine the story point value, the expert was instructed to refer to the designed requirement complexity rubric.
2. Assessment of Change Impact Requirement
   Referring to section 2.1.2, an expert is requested to assess the impact of requirement changes based on their perception. This change has the potential to cause faults, making it a valid consideration to be used as a parameter in test case prioritization. As a reference, the expert is required to refer to the provided rubric.
3. Assessment of Requirement Prioritization
   There are many requirement prioritization techniques, as described in section 2.1.3. In this study, requirement prioritization was performed by an expert through the assessment process. Generally, this prioritization is used to determine the importance of a requirement within the system, thereby dictating it priority for implementation. In this context, the prioritization level is perceived based on the importance of the requirement within the system. This implies that the greater the perceived importance of a requirement, the higher its priority not only in implementation but also during testing. To facilitate this, the research grouped the importance of requirements into five levels, namely lowest, low, medium, high, and highest.

The assessment process for the three internal factors uses the form shown in Table 3. The role of experts includes entering their assessment values for each factor across all requirements, using the provided rubric as a reference. Specifically, for the assessment of requirement complexity, they are expected to evaluate it across the three developer levels, namely junior, middle, and senior. We provide three rubric to guide experts in conducting assessments. This rubric is the result of discussions with two experts in the software industry. This assessment technique is generally used in the industry to direct software projects.

3.2.2 Assessment of Requirement Dependency

The reverse engineering was performed on the software object that serves as the case research, enabling the determination of the dependency between requirements in matrix form. The assessment conducted by experts was to determine pairwise levels according to the Saaty scale (Saaty, 2002). The reverse engineering produces a matrix, which is subsequently used to calculate the weight of dependency between requirements. The resulting dependency type are shown in Table 4.

Table 3. Assessment of Internal Factors Form



		Requirement Complexity			Change Impact of Requirement	Requirement Prioritization
		Developer				
Id	Requirements	Junior	Midle	Senior		
1	Create and Disable Patients					
2	Create, Disable, and Edit Personnel					
3	Authenticate Users					
4	Enter/Edit Demographics					
5	Log Transaction					
6	View HCP; Designate/Undesignate Licensed Health Care Professional					
7	View Access Log					
8	View Records					
9	Enter/Edit Personal Health Records					
10	Document Office Visit					
11	Determine Operational Profile					
12	Declare/Undeclare Personal Representative					
13	Maintain Standards Lists					
14	Maintain a Hospital Listing					
15	View Prescription Report					
16	View Emergency Electronic Health Record					
17	Schedule an Appointment					
18	View Comprehensive Patient Report					
19	Manage Lab Procedures					
20	Alert Users by Email					
21	View Patients					
22	Find LHCPs with Experience with a Diagnosis					
23	Messaging between LHCP and Patient					
24	View Schedule Calendar					
25	View Notifications					
26	View Activity Feed					
27	View Patient Reports					
28	Find an Expert					
29	Ward Management					
30	Basic Health Metrics					
31	View Basic Health Metrics					
32	BMI Calculation					
33	Change Password					
34	Manage Dependency					
35	Dependents' Medical Records					
36	Manage Allergy Records					

Table 4. Dependency Type (Li et al., 2012)

Weight	Dependency Type		Description
0	-	Not_Dependence	There is no dependency between requirements.
1	SM	Similar_to	A statement of one requirement is similar_to and/or overlaps with one or more others. Similar_to is used when there are different operations on the same object among similar requirements. For example, CRUD operations.
2	ST	Satisfy	One requirement indicates that the other has been fulfilled. For example, "logout" indicates that login has been successful.
3	RE	Refined_to	One requirement provides more detailed information on another requirement. In the d-model and p-model, "refines" is used to indicate the hierarchy structure of requirements. In experiments, "refines" is also used to describe requirements at the same level. For example, "set user permissions" refines "only permit users to view/edit/delete data that they have the correct permissions for." In this example, the initial requirement is explained by the final one.
4	CO	Constraint	One requirement is related to another as a constraint. For example, a "daily withdrawal limit of 10 million" is a constraint for the "withdrawal" requirement. Dependency constraints can be used to link functional and non-functional requirements.

5	PR	Precondition	One requirement is a prerequisite for the other to occur. For example, the ability to create operations is a prerequisite to the ability to delete operations. Precondition is the most common dependency type found in individual modules and relationships between modules.

Like the internal factors assessment process, the first stage in the requirement dependency assessment is determining the Saaty scale in Table 5. Next, the CW calculation is carried out and validated by calculating the CR value to see the expert's consistency in the assessment process.

To address requirement dependency weighting, experts are required to assess the PC levels for different dependency types. The process is the same as the one conducted for the PC assessment of internal factors. The difference lies in the number of items assessed in the requirement dependency, which is five, corresponding to the various types used. The instrument for conducting the PC assessment of requirement dependency is shown in Table 5.

Table 5. PC Assessment Matrix on Requirement Dependency

	PR	SM	ST	CO	RE
PR	1.00				
SM		1.00			
ST			1.00		
CO				1.00	
RE					1.00

3.3. Weighting of Internal Factors and Requirement Dependency

In TCP development, assigning weights to test cases based on the covered requirement weight requires considering both internal factors and requirement dependencies. This approach was conducted by (Abbas et al., 2019) and (Ma et al., 2016). Therefore, each requirement needs to have a weight obtained from the information on the weight of internal factors and dependency. The weighting of internal factors and dependency is stated as follows:

1. Weighting of Internal Factors

   The calculation of the weight for internal factors is performed by multiplying the CW value in the Pairwise Comparison with the value obtained from the assessment. The calculation of the weight for internal factors is formulated in equations (4), (5), and (6).

   $W_{RCX} = CW_{RCX}$ x $AssessmentValue_{RCX}$ (RCX Weight) (4)

   $W_{RCG} = CW_{RCG}$ x $AssessmentValue_{RCG.}$ (RCG Weight) (5)

   $W_{RPR} = CW_{RPR}$ x $AssessmentValue_{RPR}$ (RPR Weight) (6)

2. Weighting of Requirement Dependency

   Referring to (Vescan et al., 2017, 2021), the calculation of requirement dependency weight is based on the n x n matrix indicating the dependency types between requirements (RDR). Since both research by Vescan applied binary dependency with a values of1 and 0 depicting dependency and otherwise. Consequently, the RDR matrix in their investigation exclusively comprised 0 or 1. In this research, RDR contains the results of multiplying the Pairwise Comparison (PC) with the value of the dependency types. The calculation of RDR is carried out using formula (7).

   $RDR_{i,j} = CW_k$ x $DependencyType_{i,j}$ (7)

   The dependency weight is obtained by adding the RDR for each requirement.

3.4. Expert Identification

As perform in Krishnamoorthi study (Krishnamoorthi & Sahaaya Arul Mary, 2009), where the developer and customer assigned values to the requirement factors, we involved four experts currently working as software engineers to conduct an assessment. The purpose of engaging these experts is to ensure that the weights generated from the TCP process are based on industrial practice. It is essential to define experts by considering their experiences in the software industry, and we decided to select four categories based on the experiences. These categories are well represented by ranging from experts with over 20 years of experience to those with just 2.5 years of experience. This arrangement is necessary to obtain assessment result with a broad perspective representing four generations.

The expert selection process is conducted by identifying software companies with software engineers who meet the specified experience criteria and were ready to participate in this research. If more than one person was willing to participate at any level of experience, we selected only one person who had already read the SUT document and wanted to discuss its specifications several times. The researchers carried out this activity themselves. The detailed profiles of the experts, stressing their individual characteristics and qualifications, are shown in Table 5.

Table 5. Experts Performing the Assessment

Expert	Position in Industry	Experience in Software Engineering Industry
Exp1	Chief Operating Officer (COO) in the software industry	22 years
Exp2	Project Manager in the software industry	10 years
Exp3	Software Engineer and Scrum Master	12 years
Exp4	Junior Software Engineer	2.5 years

3.5. Software Under Test

Software under Test (SUT) used comprised three versions of iTrust and each version consisting of 36 requirements. iTrust is a widely used medical record application that has gained significant reputation from many researchers. The initial version is recognized as the original one, which is an open-source application. On the other hand, the remaining two versions, labelled A and B, have been purposefully modified to introduce variations in the number of faults and test cases for specific requirements to gain three kinds of traceability metrics. These modifications are constructed by adjusting certain test cases to identify multiple faults as well as ensuring that they are detected by more than one test case. The detailed information about the three iTrust versions is shown in Table 6.

Table 6. iTrust Versions for SUT

	Number of Req	Number of Test Cases	Number of Faults	Number of Test Cases that detect more than 1 fault	Number of Faults detected by more than 1 Test Case
Original Version	36	1499	24	2	0
Version A	36	1502	24	12	12
Version B	36	1502	24	12	16

The three versions of iTrust are described to provide information regarding the formation of tables relating to test cases versus requirements and test cases versus faults, both of which are important inputs for the TCP process. However, the assessment of requirements is sufficient to be performed on one iTrust version since all those share the same set of requirements.

4. Results and Discussion

All the stages are conducted and show the result in the form of a process model for assessing requirements parameters. Additionally, the weight of requirements is derived based on the assessment results of the parameters used.

4.1. Assessment Model

An illustration of the assessment model for requirements is shown in Fig 2. Generally, there is a standardized assessment process for both internal factors and requirement dependency. The figure clearly shows that the evaluation process for both parameters was accompanied by the PC assessment, indicating the importance level of each factor according to expert judgment. The calculated weight of internal factors and requirement dependency was subsequently used in the TCP process.

Fig 2 illustrates the assessment model for requirements. Generally, there is a standardized assessment process for both internal factors and requirement dependency. The figure clearly demonstrates that the evaluation process for both parameters was accompanied by the PC assessment, indicating the importance level of each factor according to expert judgment. The calculated weight of internal factors and requirement dependency was subsequently used in the TCP process.

Fig 3 illustrates the assessment process flow for internal factors and requirement dependencies. The process starts with a PC assessment of internal factors and requirement dependencies. Tables 7 and 8 present the assessment results by Expert-1.

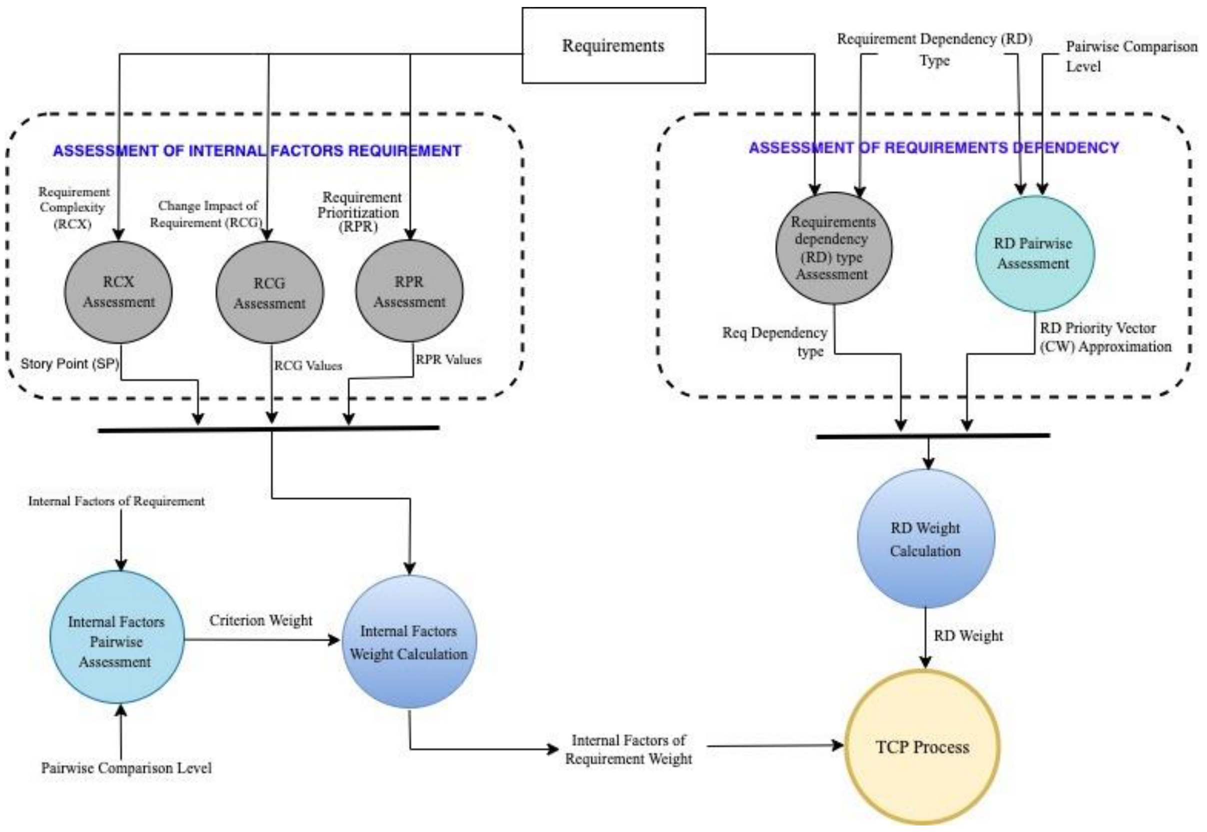

Requirements
Requirement Dependency (RD) Type
Pairwise Comparison Level
ASSESSMENT OF INTERNAL FACTORS REQUIREMENT
Requirement Complexity (RCX)
Change Impact of Requirement (RCG)
Requirement Prioritization (RPR)
RCX Assessment
RCG Assessment
RPR Assessment
Story Point (SP)
RCG Values
RPR Values
ASSESSMENT OF REQUIREMENTS DEPENDENCY
Requirements dependency (RD) type Assessment
RD Pairwise Assessment
Req Dependency type
RD Priority Vector (CW) Approximation
Internal Factors of Requirement
Internal Factors Pairwise Assessment
Criterion Weight
Internal Factors Weight Calculation
Pairwise Comparison Level
RD Weight Calculation
RD Weight
Internal Factors of Requirement Weight
TCP Process

Fig. 2: Requirement-based Assessment Model for TCP

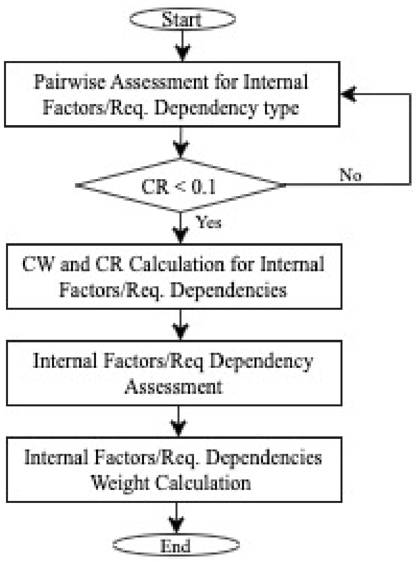

Start
Pairwise Assessment for Internal Factors/Req. Dependency type
CR < 0.1
No
Yes
CW and CR Calculation for Internal Factors/Req. Dependencies
Internal Factors/Req Dependency Assessment
Internal Factors/Req. Dependencies Weight Calculation
End

Fig. 3: Assessment Process

Table 7. Assessment Results of Internal Factors (Expert-1)

	RCX	RCG	RPR
RCX	1.00	3.00	2.00
RCG	0.33	1.00	0.33
RPR	0.50	3	1.00
Sum	1.83	7.00	3.33

Table 8. Assessment Results of Requirement Dependencies (Expert-1)

	PR	SM	ST	CO	RE
PR	1.00	9.00	3.00	1.00	3.00
SM	0.11	1.00	9.00	0.11	0.33
ST	0.33	0.11	1.00	0.50	5.00
CO	1.00	9.00	3.00	1.00	8.00
RE	0.33	3.00	0.20	0.13	1.00
Sum	2.78	22.11	16.20	2.74	17.33

After mapping the Saaty scale (Tables 7 and 8), referring to the explanation in section 3.2.1, the following calculation is used to obtain the CW and CR values for Expert-1.The stages are below:

1. Normalize Table 7 and Table 8 to obtain Table 9 and Table 10.
2. CW values for all factors were calculated so that CW values are obtained in Table 9 and Table10.

Table 9. Normalization of Internal Factors Assessment Result (based-on Table 7)

	RCX	RCG	RPR	CW
RCX	0.55	0.43	0.60	0.52
RCG	0.18	0.14	0.10	0.14
RPR	0.27	0.43	0.30	0.33

Table 10. Normalization of Req. Dependencies Assessment Result (based-on Table 8)

	PR	SM	ST	CO	RE	CW
PR	0.36	0.41	0.19	0.37	0.17	0.30
SM	0.11	1.00	9.00	0.04	0.02	0.14
ST	0.33	0.11	1.00	0.50	0.29	0.13
CO	1.00	9.00	3.00	1.00	0.46	0.36
RE	0.33	3.00	0.20	0.13	0.06	0.07

3. The results of calculating the WS and WS/CW are shown in Table 11 (for internal factors).

Table 11. Priority Vector Validation of Internal Factors

	RCX	RCG	RPR	CW	WS	WS/CW
RCX	0.52	0.42	0.67	0.52	1.62	3.08
RCG	0.17	0.14	0.11	0.14	0.43	3.02
RPR	0.26	0.42	0.33	0.33	1.02	3.06

3. Calculate the λ_max value, using formula (1):

$$\lambda_{max} = \frac{3.08 + 3.02 + 3.06}{3} = 3.05$$

4. Calculate the CI using formula (2):

$$CI = \frac{\lambda_{max} - n}{n-1} = \frac{3.05 - 3}{3-1} = 0.027$$

   In the internal factors case, the number of criteria observed is 3, n=3, and the RI value is 0.58 (Table 2).
5. The last is the calculation of the CR value, using formula (3).

$$CR = \frac{CI}{RI} = \frac{0.027}{0.58} = 0.046$$

According to Saaty, if the CR value is <10%, this value can be accepted as a weight reference. On the other hand, if the CR value is >= 10% (0.1), then the value is inconsistent and cannot be used as a weight reference. In the same way, the above process can be applied to determining the consistency ratio (CR) value for requirements dependencies assessment. Furthermore, experts must refer to the rubric in Tables 13, 14, and 15 when assessing internal factors. Meanwhile, to carry out a requirements dependency assessment, refer to Table 4.

To assess requirement complexity, an expert considered the implementation effort. For instance, assuming a requirement can be implemented by a junior developer in less than two hours, the expert assigns a story point value of one, as outlined in Table 3. On the contrary, assuming a senior developer anticipates that a challenging requirement would take 15 hours, the expert assigns a story point value of 8, etc. While it remains subjective, this framework provided guidelines for determining the story point value of a requirement.

Table 13. Requirement Complexity Estimation

Fibonacci values	Requirement Complexity Estimation
1	Easy, can be completed in < 2 hours.
2	Easy, can be completed in 3 to < 4 hours.
3	Medium can be completed in 4 to < 8 hours.
5	Medium can be completed in 8 to < 12 hours.
8	Medium can be completed in 12 to < 16 hours.
13	Difficult, can be completed in 16 to < 24 hours.
21	Difficult, can be completed in 24 to < 32 hours.
34	Difficult, can be completed in > 32 hours.

Table 14. Weight of Change Impact Requirement

Weight	Description
1	If a requirement changes, it does not impact others.
2	If a requirement changes, it might have a minimal impact on others.
3	If a requirement changes, it could affect others but not significantly impact the overall system. For instance, adding a feature for 2 languages (Language A and B) would require updates across requirements, but it would not affect the system's performance.
4	If a requirement changes, it impacts many others. For example, modifying the feature for Alerting Stock of Medicine requires checking stock and other requirements.
5	If a requirement changes, it has a serious impact on many others and significantly affects the entire system. For example, a requirement change necessitates changes in data structure or technology.

Table 15. Weighting for Requirement Prioritization

Value	Description
1	Lowest: The requirement is not important in the system
2	Low: The requirement plays a somewhat unimportant role in the system
3	Medium: The requirement plays a moderately important role in the system
4	High: The requirement plays an important/urgent role in the system
5	Highest: The requirement plays a very important/urgent/critical role in the system

4.2. Assessment Results

This section summarizes the assessment results from the four experts. Table 16 displays the assessment results of internal factors to represent the results from Expert-1 and Expert-2. The CW and CR values obtained during the CP assessment stage for internal factors are shown in Table 17. Furthermore, the results of reverse engineering applied to iTrust to obtain the dependency matrix, and the assessment results for PC of dependency types are shown in Tables 18 and 19, respectively.

Table 16. Assessment Results of Internal Factors (Expert-1 and Expert-2)

#Req	**Expert-1**					**Expert-2**				
	RCX			**RCG**	**RPR**	**RCX**			**RCG**	**RPR**
	Junior	Middle	Senior			Junior	Middle	Senior		
1	3	2	2	4	5	21	13	8	5	5
2	5	3	3	4	4	8	5	3	5	5
3	2	1	1	5	4	3	2	1	2	4
4	3	2	2	3	3	8	5	3	3	5
:	:	:	:	:	:	:	:	:	:	:
:	:	:	:	:	:	:	:	:	:	:
35	3	2	2	3	3	5	3	1	2	1
36	5	3	3	4	3	21	8	5	4	5

Table 17 presents the criteria weight (CW) and consistency ratio (CR) values for internal factors provided by the four experts. These values are obtained based on the assessment results and subsequent calculations using the Saaty method. According to the four experts, the CR ranges from 0.02 to 0.08, with two of them sharing identical CR values, indicating a close association in their perceptions regarding the importance of the three internal factors. The CR values in the table are considered valid as these are less than 0.1.

Table 17. CW and CR Values for Internal Factors

Expert	CW			CR
	RCX	RCG	RPR	
Expert-1	0.52	0.14	0.33	0.05
Expert-2	0.6	0.17	0.23	0.08
Expert-3	0.6	0.17	0.23	0.08
Expert-4	0.65	0.25	0.1	0.02

The assessment results for the PC of requirement dependency are shown in Table 18. These results indicate the importance level of dependency types as evaluated by the four experts. According to the four experts, the CR ranges from 0.02 to 0.08. Despite the distribution is not being very close, all CR values were less than 0.1 therefore, the assessment results are consistent and valid for use in the subsequent stages. Considering the CR value, the consistency experts' assessments of internal factors and requirement dependencies are explained below: Expert-3 and Expert-4 are more consistent than Expert-1 and Expert-2. In assessing internal factors, the CR values from Expert-1 to Expert-4 are 0.05, 0.08, 0.08, and 0.02, respectively, while in the requirements dependency assessment, the CR values are 0.02, 0.04, 0.08, and 0.03. With four experts in this study, the CR value is not significantly related to experience. In the internal factors assessment, Expert-1 and Expert-4 have closer CR values, compared to Expert-2 and Expert-3, even though the difference in experience between Expert-1 and Expert-4 is further, compared to Expert-2 and Expert-3. This is another issue worth researching if more experts are involved and the factors studied differ.

Table 18. Results of Requirement Dependency Assessment

#Expert	CW					CR
	PR	SM	ST	CO	RE	
Expert-1	0.37	0.04	0.14	0.37	0.08	0.02
Expert-2	0.42	0.04	0.07	0.42	0.05	0.04
Expert-3	0.18	0.51	0.06	0.18	0.07	0.08
Expert-4	0.35	0.08	0.07	0.35	0.15	0.03

Meanwhile the requirement dependency matrix for iTrust is shown in Table 19. The diagonal elements containing '-' signify the dependency from requirement *n* x *n*. Note: coloured items indicate dependency between requirements, with dependency type referring to the Li et al. model (Li et al., 2012).

Table 19. Requirement Dependency Matrix on iTrust

# Requirement	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	..	..	35	36
1	-	0	5	0	0	0	0	0	1	0	0	0	0	0	0			5	5
2	0	-	5	0	0	1	0	0	0	0	0	0	0	0	0			0	0
3	0	0	-	0	0	0	0	0	0	0	0	0	0	0	0			0	0
4	0	0	5	-	0	0	0	0	0	0	0	0	0	0	0			0	0
5	0	0	5	0	-	0	0	0	0	0	0	0	0	0	0			0	0
6	0	1	5	0	0	-	0	0	0	0	0	0	0	0	0			0	0
7	5	0	5	0	0	0	-	0	0	0	0	0	0	0	0			0	0
8	0	0	5	0	0	0	0	-	0	0	0	0	0	0	0			0	0
9	1	0	5	0	0	0	0	0	-	0	0	0	0	0	0			0	0
10	0	0	5	0	0	0	0	0	0	-	0	0	0	0	0			0	0
11	0	0	5	0	0	0	0	0	0	0	-	0	0	0	0			0	0
12	2	0	5	0	0	0	0	0	0	0	0	-	0	0	0			0	0
13	0	2	5	0	0	0	0	0	0	0	0	0	-	0	0			0	0
14	0	0	5	0	3	0	0	0	0	0	0	0	0	-	0			0	0
15	0	0	5	0	3	0	0	0	0	0	0	0	0	0	-			0	0
:	:	:	:	:	:	:	:	:	:	:	:	:	:	:	:	-	:	:	:
:	:	:	:	:	:	:	:	:	:	:	:	:	:	:	:	:	-	:	:
35	3	0	5	0	3	0	0	0	0	0	0	0	0	0	0			-	0
36	0	0	5	0	0	0	0	0	0	0	0	0	0	0	0			0	-

Previous requirements-based studies used assessment results that involved several parties. For instance, in the Krishnamoorthy's study (Krishnamoorthi & Sahaaya Arul Mary, 2009) customers and developers conducted a process to provide value. The value, which represents the complexity factor for implementation, ranges from 1 to 10. The 1 value indicates the lowest complexity, while 10 represents the highest complexity. The values entered are based on the judgment of customers or developers, there were no detailed explanation regarding the characteristics or value criteria in this range. This study presents some rubrics to guide the expert in the assessment process.

On the other hand, in this study, pairwise comparison is used to determine the level of importance of a factor compared to other factors. The weight of a factor is determined not only by the assessment values but also by its relative importance. Additionally, pairwise comparison ensures that expert assessments are consistent. If the assessment values are inconsistent, experts are asked to conduct iterations until consistency is achieved. According to Saaty (Saaty, 2002), the CW values are considered valid if the CR is less than 0.1.

4.3. Requirement Weight Calculation

The weight of internal factors is calculated by multiplying the CW values from the Pairwise Comparison with the assessment results (Formulas 1, 2, and 3). Table 20 presents the internal factors weight for Expert-1 and Expert-2.

Table 20. Weight of Internal Factors (Expert-1 and Expert-2)

#Req	Expert-1					Expert-2				
	RCX			RCG	RPR	RCX			RCG	RPR
	Junior	Middle	Senior			Junior	Middle	Senior		
1	1.56	1.04	1.04	0.56	1.65	12.62	7.81	4.81	0.85	1.14
2	2.6	1.56	1.56	0.56	1.32	4.81	3.00	1.80	0.85	1.14
3	1.04	0.52	0.52	0.7	1.32	1.80	1.20	0.60	0.34	0.92
4	1.56	1.04	1.04	0.42	0.99	4.81	3.00	1.80	0.51	1.14
:	:	:	:	:	:	:	:	:	:	:
:	:	:	:	:	:	:	:	:	:	:
35	1.56	1.04	1.04	0.42	0.99	3.00	1.80	0.60	0.34	0.23
36	2.6	1.56	1.56	0.56	0.99	12.62	4.81	3.00	0.68	1.14

The weight for requirement dependency is obtained by initially creating the RDR, which contains the values resulting from multiplying the dependency type matrix by the CW from the Pairwise Comparison, (Formula 4). The RDR is represented as an *n* x *n* matrix, illustrating the presence or absence of dependency and its associated types. An example of the RDR matrix from Expert-2 is shown in Table 21.

Table 21. RDR Matrix (Expert-2)

# Requirement	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	..	..	35	36
1	-	0	2.48	0	0	0	0	0	0.09	0	0	0	0	0	0			2.48	2.48
2	0	-	2.48	0	0	0.09	0	0	0	0	0	0	0	0	0			0	0
3	0	0	-	0	0	0	0	0	0	0	0	0	0	0	0			0	0
4	0	0	2.48	-	0	0	0	0	0	0	0	0	0	0	0			0	0
5	0	0	2.48	0	-	0	0	0	0	0	0	0	0	0	0			0	0
6	0	0.09	2.48	0	0	-	0	0	0	0	0	0	0	0	0			0	0
7	2.48	0	2.48	0	0	0	-	0	0	0	0	0	0	0	0			0	0
8	0	0	2.48	0	0	0	0	-	0	0	0	0	0	0	0			0	0
9	0.09	0	2.48	0	0	0	0	0	-	0	0	0	0	0	0			0	0
10	0	0	2.48	0	0	0	0	0	0	-	0	0	0	0	0			0	0
11	0	0	2.48	0	0	0	0	0	0	0	-	0	0	0	0			0	0
12	0.12	0	2.48	0	0	0	0	0	0	0	0	-	0	0	0			0	0
13	0	0.12	2.48	0	0	0	0	0	0	0	0	0	-	0	0			0	0
14	0	0	2.48	0	0.15	0	0	0	0	0	0	0	0	-	0			0	0
15	0	0	2.48	0	0.15	0	0	0	0	0	0	0	0	0	-			0	0
:	:	:	:	:	:	:	:	:	:	:	:	:	:	:	:	-	:	:	:
:	:	:	:	:	:	:	:	:	:	:	:	:	:	:	:	:	-	:	:
35	0.15	0	2.48	0	0.15	0	0	0	0	0	0	0	0	0	0			-	0
36	0	0	2.48	0	0	0	0	0	0	0	0	0	0	0	0			0	-

Furthermore, the dependency weight is calculated by summing the weight values for each requirement, as expressed in each row of the RDR matrix. Cumulative sum results obtained by the four experts are shown in Tables 22.

Table 22. Results of Requirement Dependency Weight

#Expert	#Requirement																	
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
Expert-1	7.50	1.89	0.04	1.86	1.86	2.13	3.71	1.86	1.93	1.86	1.86	2.60	2.14	3.94	2.09	1.86	5.06	2.37
Expert-2	10.09	2.57	0.09	2.48	2.48	2.72	4.95	2.48	2.66	2.48	2.48	2.90	2.60	5.10	2.63	2.48	5.04	2.75
Expert-3	4.63	1.41	0.51	0.90	0.90	1.62	1.80	0.90	1.93	0.90	0.90	1.42	1.02	2.00	1.10	0.90	2.55	1.22
Expert-4	7.20	1.84	0.08	1.76	1.76	2.28	3.52	1.76	1.93	1.76	1.76	2.77	1.89	3.95	2.20	1.76	5.01	2.33

	#Requirement																	
	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36
Expert-1	1.86	2.46	1.93	3.71	2.71	2.14	2.46	2.09	1.86	2.43	2.43	2.37	2.60	2.09	0.04	0.00	2.78	1.86
Expert-2	2.48	2.81	2.66	4.95	2.84	2.60	2.81	2.63	2.48	2.72	2.72	2.75	2.90	2.63	0.09	0.00	3.07	2.48
Expert-3	0.90	1.65	1.93	1.80	1.26	1.02	1.65	1.10	0.90	1.14	1.14	1.22	1.42	1.10	0.51	0.00	1.71	0.90
Expert-4	1.76	2.11	1.93	3.52	2.15	1.89	2.11	2.20	1.76	2.02	2.02	2.33	2.77	2.20	0.08	0.00	3.51	1.76

This assessment model allows for a more in-depth study involving several SUTs with more varied requirements and characteristics, for example, by testing smaller software (fewer requirements and dependencies) or vice versa.

5. Conclusion

This paper presents a novel assessment model for requirement-based test case prioritization (TCP) that integrates internal factors and requirement dependencies. The model employs pairwise comparison and expert evaluation techniques to assign quantitative weights to the factors, providing a more objective and nuanced basis for designing TCP strategies. The assessment process was validated using the iTrust medical record system as a case study, with four experts from diverse software engineering backgrounds participating in the evaluation.

The proposed model makes several contributions to the field of software testing and agile development. First, it offers a comprehensive framework for assessing requirement factors that goes beyond existing approaches by considering both internal attributes and dependencies. Second, it demonstrates the feasibility and value of incorporating expert judgment into the assessment process, which can help to capture context-specific knowledge and priorities. Third, it provides a practical tool for software practitioners seeking to optimize their TCP processes and improve testing effectiveness.

However, the study also has some limitations that should be acknowledged. The small sample size of experts and the potential subjectivity of their assessments may limit the generalizability of the findings. Future research could address these issues by recruiting a larger and more diverse panel of experts and employing more rigorous validation techniques, such as inter-rater reliability analysis or experimental studies comparing the effectiveness of different assessment models.

In conclusion, this paper makes a valuable contribution to the growing body of research on requirement-based TCP and offers a promising approach for enhancing testing efficiency and effectiveness in agile software development. The proposed assessment model provides a foundation for future work in this area and has the potential to be adapted and extended to different software development contexts and methodologies. Software practitioners and researchers alike can benefit from the insights and techniques presented in this study as they seek to optimize their TCP processes and deliver high-quality software products.

Acknowledgments

The authors express their gratitude to PT. Padepokan Tujuh Sembilan Bandung, Indonesia, for funding the publication of this research.

References

Abbas, M., Inayat, I., Saadatmand, M., & Jan, N. (2019). Requirements Dependencies-Based Test Case Prioritization for Extra-Functional Properties. *2019 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW)*, 159–163. https://doi.org/10.1109/ICSTW.2019.00045

Akbar, M. A., Naveed, W., Alsanad, A. A., Alsuwaidan, L., Alsanad, A., Gumaei, A., Shafiq, M., & Riaz, M. T. (2020). *Requirements Change Management Challenges of Global Software Development : An Empirical Investigation*. *8*. https://doi.org/10.1109/ACCESS.2020.3035829

Apurva, A. (2021). Complexity Estimation for Distributed Software Development Using SRS. *IOP Conference Series: Materials Science and Engineering*, *1116*(1), 012193. https://doi.org/10.1088/1757-899x/1116/1/012193

Borhan, N. H., Zulzalil, H., Hassan, S., & Ali, N. M. (2019). Requirements prioritization techniques focusing on agile software development: A systematic literature review. *International Journal of Scientific and Technology Research*, *8*(11), 2118–2125.

Cardoso, J. (2014). *About the Complexity of Teamwork and Collaboration Processes*. *May*. https://doi.org/10.1109/SAINTW.2005.1620015

Carlshamrea, P., Sandahla, K., Lindvallb, M., Regnellc, B., & Natt, J. (2001). *An Industrial Survey of Requirements Interdependencies in Software Product Release Planning*.

Chi, J., Qu, Y., Zheng, Q., Yang, Z., Jin, W., Cui, D., Liu, T., Qu, Y., Zheng, Q., Yang, Z., Jin, W., Cui, D., & Liu, T. (2020). Relation-based Test Case Prioritization for Regression Testtest case prioritization for regression testing. *The Journal of Systems & Software*. https://doi.org/10.1016/j.jss.2020.110539

Dahlstedt, Å. G., & Persson, A. (2005). Requirements interdependencies: State of the art and future challenges. *Engineering and Managing Software Requirements*, 95–116. https://doi.org/10.1007/3-540-28244-0_5

Deshpande, G., Arora, C., & Ruhe, G. (2019). Data-driven elicitation and optimization of dependencies between requirements. *Proceedings of the IEEE International Conference on Requirements Engineering*, *2019-Septe*(iii), 416–421. https://doi.org/10.1109/RE.2019.00055

Deshpande, G., Motger, Q., Palomares, C., Kamra, I., Biesialska, K., Franch, X., Ruhe, G., & Ho, J. (2020). Requirements Dependency Extraction by Integrating Active Learning with Ontology-Based Retrieval. *Proceedings of the IEEE International Conference on Requirements Engineering*, *2020-Augus*, 78–89. https://doi.org/10.1109/RE48521.2020.00020

Di Nucci, D., Panichella, A., Zaidman, A., & De Lucia, A. (2020). A Test Case Prioritization Genetic Algorithm Guided by the Hypervolume Indicator. *IEEE Transactions on Software Engineering*, *46*(6), 674–696. https://doi.org/10.1109/TSE.2018.2868082

Fernández-Diego, M., Méndez, E. R., González-Ladrón-De-Guevara, F., Abrahão, S., & Insfran, E. (2020). An update on effort estimation in agile software development: A systematic literature review. *IEEE Access*, *8*, 166768–166800. https://doi.org/10.1109/ACCESS.2020.3021664

Habtemariam, G. M., & Mohapatra, S. K. (2019). *A Genetic Algorithm-Based Approach for Test Case Prioritization*. Springer International Publishing. https://doi.org/10.1007/978-3-030-26630-1

Hasnain, M., Pasha, M. F., Ghani, I., & Jeong, S. R. (2021). Functional Requirement-Based Test Case Prioritization in Regression Testing: A Systematic Literature Review. In *SN Computer Science* (Vol. 2, Issue 6). Springer. https://doi.org/10.1007/s42979-021-00821-3

Hemmati, H. (2019). Advances in Techniques for Test Prioritization. In *Advances in Computers* (1st ed., Vol. 112). Elsevier Inc. https://doi.org/10.1016/bs.adcom.2017.12.004

Hettiarachchi, C., & Do, H. (2019). A Systematic Requirements and Risks-Based Test Case Prioritization Using a Fuzzy Expert System. *Proceedings - 19th IEEE International Conference on Software Quality, Reliability and Security, QRS 2019*, 374–385. https://doi.org/10.1109/QRS.2019.00054

Hettiarachchi, C., Do, H., & Choi, B. (2014). Effective regression testing using requirements and risks. *Proceedings - 8th International Conference on Software Security and Reliability, SERE 2014*, 157–166. https://doi.org/10.1109/SERE.2014.29

Hettiarachchi, C., Do, H., & Choi, B. (2016). Risk-based test case prioritization using a fuzzy expert system. *Information and Software Technology*, *69*, 1–15. https://doi.org/10.1016/j.infsof.2015.08.008

Hudaib, A., Masadeh, R., Qasem, M. H., & Alzaqebah, A. (2018). Requirements Prioritization Techniques Comparison. *Modern Applied Science*, *12*(2), 62. https://doi.org/10.5539/mas.v12n2p62

Hujainah, F., Bakar, R. B. A., Abdulgabber, M. A., & Zamli, K. Z. (2018). Software Requirements Prioritisation: A Systematic Literature Review on Significance, Stakeholders, Techniques and Challenges. *IEEE Access*, *6*, 71497–71523. https://doi.org/10.1109/ACCESS.2018.2881755

Khatibsyarbini, M, Isa, M. A., Jawawi, D. N. A., Hamed, H. N. A., & Suffian, M. D. M. (2019). Test Case Prioritization Using Firefly Algorithm for Software Testing. *IEEE Access*, *7*, 132360–132373. https://doi.org/10.1109/ACCESS.2019.2940620

Khatibsyarbini, Muhammad, Isa, M. A., Jawawi, D. N. A., & Tumeng, R. (2018). Test case prioritization approaches in regression testing: A systematic literature review. *Information and Software Technology*, *93*(June 2018), 74–93. https://doi.org/10.1016/j.infsof.2017.08.014

Krishnamoorthi, R., & Sahaaya Arul Mary, S. A. (2009). Requirement based system test case prioritization of new and regression test cases. *International Journal of Software Engineering and Knowledge Engineering*, *19*(3), 453–475. https://doi.org/10.1142/S0218194009004222

Kulasinghe, Y. (2021). *Using Story Point for Effort Estimation in Agile Software Development in Time and Cost*. *November*. https://doi.org/10.13140/RG.2.2.12898.25281

Li, J., Zhu, L., Jeffery, R., Liu, Y., Zhang, H., Wang, Q., & Li, M. (2012). An initial evaluation of requirements dependency types in change propagation analysis. *IET Seminar Digest*, *2012*(1), 62–71. https://doi.org/10.1049/ic.2012.0009

Ma, T., Zeng, H., & Wang, X. (2016). Test case prioritization based on requirement correlations. *2016 IEEE/ACIS 17th International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing, SNPD 2016*, *61170044*, 419–424. https://doi.org/10.1109/SNPD.2016.7515934

Mishra, D. B., Mishra, R., Acharya, A. A., & Das, K. N. (2019). Test case optimization and prioritization based on multi-objective genetic algorithm. In *Advances in Intelligent Systems and Computing* (Vol. 741). Springer Singapore. https://doi.org/10.1007/978-981-13-0761-4_36

Noviyanto, F., Razali, R., & Nazri, M. Z. A. (2023). Understanding requirements dependency in requirements prioritization: a systematic literature review. *International Journal of Advances in Intelligent Informatics*, *9*(2), 249–272. https://doi.org/10.26555/ijain.v9i2.1082

Oswaldo, M. I., Saikhu, A., & Amaliah, B. (2014). Implementasi Metode Pairwise Comparison Pada Uji Kinerja Varian Metode Kecerdasan Buatan Pada Penyelesaian Masalah TSP. *Teknik POMITS*, *2*(1), 1–6.

Prado Lima, J. A., & Vergilio, S. R. (2020). Test Case Prioritization in Continuous Integration environments: A systematic mapping study. *Information and Software Technology*, *121*, 106268. https://doi.org/10.1016/j.infsof.2020.106268

Rahmani, A., Ahmad, S., Jalil, I. E. A., & Herawan, A. P. (2021). A Systematic Literature Review on Regression Test Case Prioritization. *I(IJACSA) International Journal of Advanced Computer Science and Applications*, *12*(9), 253–267. https://doi.org/10.14569/IJACSA.2021.0120929

Saaty, T. L. (2002). Decision making with the Analytic Hierarchy Process. *Scientia Iranica*, *9*(3), 215–229. https://doi.org/10.1504/ijssci.2008.017590

Saher, N. (2017). *Requirement Change Taxonomy and Categorization in Agile Software Development*.

Salmanoglu, M., Hacaloglu, T., & Demirors, O. (2017). Effort estimation for agile software development: Comparative case studies using COSMIC functional size measurement and story points. *ACM International Conference Proceeding Series*, *Part F1319*(November), 41–49. https://doi.org/10.1145/3143434.3143450

Samad, A., Mahdin, H. Bin, Kazmi, R., Ibrahim, R., & Baharum, Z. (2021). *Multiobjective Test Case Prioritization Using Test Case Effectiveness : Multicriteria Scoring Method*. *2021*.

Scott, E., & Pfahl, D. (2018). Using developers' features to estimate story points. *ACM International Conference Proceeding Series*, *106*, 106–110. https://doi.org/10.1145/3202710.3203160

Sharif, A., Marijan, D., & Liaaen, M. (2021). DeepOrder: Deep Learning for Test Case Prioritization in Continuous Integration Testing. *Proceedings - 2021 IEEE International Conference on Software Maintenance and Evolution, ICSME 2021*, 525–534. https://doi.org/10.1109/ICSME52107.2021.00053

Silvarajoo, A., & Hazim Alkawaz, M. (2022). A Framework for Optimizing Software Regression Test Case based on Modified-Ant Colony Optimization (M-ACO). *ITM Web of Conferences*, *42*, 01007. https://doi.org/10.1051/itmconf/20224201007

Srikanth, H., Hettiarachchi, C., & Do, H. (2016). Requirements based test prioritization using risk factors: An industrial study. *Information and Software Technology*, *69*, 71–83. https://doi.org/10.1016/j.infsof.2015.09.002

Srikanth, H., Williams, L., Osborne, J., Science, C., Carolina, N., & Carolina, N. (2005). *System Test Case Prioritization of New and Regression Test Cases 3 . Prioritizations of Requirements for Testing*.

Tawosi, V., Al-Subaihin, A., & Sarro, F. (2022). Investigating the Effectiveness of Clustering for Story Point Estimation. *Proceedings - 2022 IEEE International Conference on Software Analysis, Evolution and Reengineering, SANER 2022*, 827–838. https://doi.org/10.1109/SANER53432.2022.00101

Vescan, A., Chisalita-Cretu, C., Serban, C., & Diosan, L. (2021). On the use of evolutionary algorithms for test case prioritization in regression testing considering requirements dependencies. In *AISTA 2021 - Proceedings of the 1st ACM International Workshop on AI and Software Testing/Analysis, co-located with ECOOP/ISSTA 2021* (Vol. 1, Issue 1). Association for Computing Machinery. https://doi.org/10.1145/3464968.3468407

Vescan, A., Şerban, C., Chisăliţă-Creţu, C., & Dioşan, L. (2017). Requirement dependencies-based formal approach for test case prioritization in regression testing. *Proceedings - 2017 IEEE 13th International Conference on Intelligent Computer Communication and Processing, ICCP 2017*, *March*, 181–188. https://doi.org/10.1109/ICCP.2017.8117002

Yaraghi, A. S., Bagherzadeh, M., Kahani, N., & Briand, L. C. (2023). Scalable and Accurate Test Case Prioritization in Continuous Integration Contexts. *IEEE Transactions on Software Engineering*, *49*(4), 1615–1639. https://doi.org/10.1109/TSE.2022.3184842

Yoon, M., Lee, E., Song, M., & Choi, B. (2012). A Test Case Prioritization through Correlation of Requirement and Risk. *Journal of Software Engineering and Applications*, *05*(10), 823–835.

https://doi.org/10.4236/jsea.2012.510095

Zhang, H., Li, J., Zhu, L., Jeffery, R., Liu, Y., Wang, Q., & Li, M. (2014). Investigating dependencies in software requirements for change propagation analysis. *Information and Software Technology*, *56*(1), 40–53. https://doi.org/10.1016/j.infsof.2013.07.001