Computers, Materials & Continua
DOI:10.32604/cmc.2021.014830
Article

Tech Science Press

Role of Fuzzy Approach towards Fault Detection for Distributed Components

Yaser Hafeez[1], Sadia Ali[1], Nz Jhanjhi[2], Mamoona Humayun[3], Anand Nayyar[4,5,*] and Mehedi Masud[6]

[1]University Institute of Information Technology, Pir Mehr Ali Shah Arid Agriculture University, Rawalpindi, Pakistan
[2]School of Computer Science and Engineering, SCE, Taylor's University, Subang Jaya, Malaysia
[3]Department of Information systems, College of Computer and Information Sciences, Jouf University, Saudi Arabia
[4]Graduate School, Duy Tan University, Da Nang, 550000, Viet Nam
[5]Faculty of Information Technology, Duy Tan University, Da Nang, 550000, Viet Nam
[6]Department of Computer Science, College of Computers and Information Technology, Taif University, Taif, 21944, Saudi Arabia
[*]Corresponding Author: Anand Nayyar. Email: anandnayyar@duytan.edu.vn
Received: 20 October 2020; Accepted: 12 December 2020

Abstract: Component-based software development is rapidly introducing numerous new paradigms and possibilities to deliver highly customized software in a distributed environment. Among other communication, teamwork, and coordination problems in global software development, the detection of faults is seen as the key challenge. Thus, there is a need to ensure the reliability of component-based applications requirements. Distributed device detection faults applied to tracked components from various sources and failed to keep track of all the large number of components from different locations. In this study, we propose an approach for fault detection from component-based systems requirements using the fuzzy logic approach and historical information during acceptance testing. This approach identified error-prone components selection for test case extraction and for prioritization of test cases to validate components in acceptance testing. For the evaluation, we used empirical study, and results depicted that the proposed approach significantly outperforms in component selection and acceptance testing. The comparison to the conventional procedures, i.e., requirement criteria, and communication coverage criteria without irrelevancy and redundancy successfully outperform other procedures. Consequently, the F-measures of the proposed approach define the accurate selection of components, and faults identification increases in components using the proposed approach were higher (i.e., more than 80 percent) than requirement criteria, and code coverage criteria procedures (i.e., less than 80 percent), respectively. Similarly, the rate of fault detection in the proposed approach increases, i.e., 92.80 compared to existing methods i.e., less than 80 percent. The proposed approach will provide a comprehensive guideline and roadmap for practitioners and researchers.

Keywords: Component-based software; selection; acceptance testing; fault detection

1 Introduction

Component-based software (CBS) is used in daily life everywhere viz: Multimedia, offices, household items, etc., [1,2] and Internet of Things applications to improve medical, transportation, military, etc. based sectors using advance artificial intelligent technology [3]. Due to complexity, agility to deliver, and multiuser satisfaction; product development is divided into different components and provide customization option to the similar multi-user of CBS according to their preferences. Customization in CBS is the combination of different service arrangements to facilitate a similar multi-user in one product, e.g., Microsoft Office, operating systems, mobile, etc. Accordingly, the development team reuses most of the customized options from the previous version or other CBS applications. The customized options for CBS selected from different sources viz: it may be a built-in organization, it may be purchased from outsources, and used open sources components [4,5]. Hence, the CBS components are mostly developed by a third party, and code information is not available. CBS developed based on a component-based software engineering approach, which is the reuse and encapsulation approach. Thus, CBS is the combination of loosely coupled independent components with specific functionality interfaces [2,6–8]. Due to reusability, encapsulating, and multi-sourcing of components; different issues were created during the integration of these components. As, if there are ambiguities and incompleteness among requirements of CBS during selection and prioritization of components then it causes failure in the system. Components requirements are the main important element for developing and managing components from specification to verification [2,4,5,9,10]. Thus, ambiguities and incompleteness occur during elicitation of requirements from different globally distributed stakeholders due to ignorance of the multi-perspective of stakeholders. As, components for reuse available at different sources, i.e., in an organization, outsource, open-source, and off-the-shelf components [4]. Therefore, the distributed location of stakeholders and components creates challenges, for the reuse of components while a selection from multiple distributed sources with the highest accurate integration to build a new CBS application.

The development in a distributed environment fulfills the demand for multi-perspective stakeholders according to modern technology. So, global software development (GSD) provides a paradigm to build in a distributed environment, where resources, stakeholders, and project teams are located at different locations all over the world [11,12]. Thus, GSD helped to share knowledge, resources, skills, and decrease cost with an increase in user satisfaction and quality. Although, there are some risks of communication, control, and coordination among stakeholders, project teams, and development activities synchronization. Due to different times, customs, languages, and geographical conditions [7,12]. Hence, language, distribution, and multi-perspective create challenges of semantic, ambiguities, and incompletes highlighted by existing practices during requirements specification and reuse of CBS requirements during development. Thus, in the case of CBS, if components are not accurately identified due to distributed resources, stakeholders, and distributed component sources, it affects CBS quality after integration and decreases user satisfaction [8,9,12]. Subsequently, during validation of components requirements in acceptance testing phases system fails and CBS cannot put into operation. This is due to the wrong identification of components' functionality and selection of components for identification.

Acceptance testing aims to ensure that integration of all components as a single system meets all its functionalities without any error [6,8,12]. In acceptance testing of CBS development, all the components are validated according to user satisfaction before handover to stakeholders. Acceptance testing involves different methods, i.e., prioritization, selection, and minimization to identified error-prone test cases. A large set of test cases are used to validate and most of them

reuse due to the reusability property of CBS [6,12]. Therefore, the required appropriate criteria for error-prone test case identification to detect maximum faults and reduce execution time during validation. Most of the existing practices and adopted prioritization methods do not reduce the size of test cases, deleted test cases permanently, and not created irrelevancy among test cases. So, prioritization method sort all test cases according to their error occurrence chances using different criteria i.e., time redundancy and spatial redundancy in CBS which only deals with hardware validation and ignore software validation. For software faults detection few practices available based on prioritizing software test cases using prioritization criteria, i.e., requirements dependency, code coverage, history-based, etc. History-based criteria are considered as important and frequently adopted criteria for prioritization of test cases due to the use of historical information [13,14]. Historical information consists of faults detection rate, execution rate, change in code, change in test cases, requirements changes bug report, etc., from previous and similar domain components. Thus, to reduce failure and increase the faults detection rate for acceptance testing of CBS multi-perspective of user requirements. As requirements are the main element of every software development, which depend on user demands to fulfill modern era business and daily life requirements. There is a need for accurate and correct selection and prioritization of components for reuse from distributed sources.

The main objective of this research work is to propose an intelligent approach for components selection, and prioritization of test cases in GSD for acceptance testing of multi-perspective stakeholders without any ambiguities, irrelevancy, redundancy, and communication issues. Thus, for components selection, we used the fuzzy approach, and the fuzzy approach is the method of providing logic and reasons for ambiguous requirements. Thus, the fuzzy approach is the best for managing inconsistent and ambiguous component requirements and multi-criteria problems during components selection [12,15]. Similarly, for acceptance testing, we used historical information, i.e., frequently failed and reused components test cases for validating CBS. As the history-based approach is the maintenance of CBS previous versions information from CBS specification to validation process like requirements information, code information, test execution, etc. information which helped in CBS reuse specifically in GSD development [13,14]. In this paper, our main contributions are as follow:

- We present components selection and acceptance testing proposed approach using fuzzy methods and historical information criteria on multi-perspective stakeholders and less involvement of users in GSD.
- The fuzzy method used for resolving components selection and prioritization to resolve semantic and term mismatch issues during reusability of CBS due to diverse stakeholder perspective intelligently. While for acceptance testing defined history-based test case prioritization criteria for higher user components software acceptance.
- Therefore, our approach resolves the drawback of CBS selection and acceptance testing. The proposed approach was evaluated using an experimental study on industrial projects of GSD for demonstrating our proposed approach.
- The proposed approach compared with existing conventional procedures and results depicted significant improvement in components selection and acceptance testing in GSD.
- And the study provides an empirical suggestion for future investigation in the domain of CBS development and validation.

We organized the remaining paper as follows: Section 2 describes relevant studies review and critical analysis of literature to identified prevailing challenges. Then in Section 3, we described our proposed approach procedure to mitigate challenges. We evaluated our approach on industrial

projects in Section 4 to illustrate their results and discussion. Finally, we conclude our research findings and describe future research in Section 5.

2 Related Work

In this section, we provide background about GSD and CBS development and current challenges in existing studies we describe some existing relevant studies. Several studies mentioned GSD challenges and presented different solutions in the existing literature. Therefore, Shameem et al. [12] presented a prioritization process with success factors taxonomy in GSD for scaling agile methodology and in [16] provide challenges of agile method development in GSD taxonomical classification. Similarly, in author presented the current state of practices in GSD during agile method scaling. The impact of changes in requirements during GSD product development described in [17] and in [18] provided a roadmap to improve software development in GSD. While in [19] authors supported the concept of reuse requirements automatically in GSD. All these existing studies [20–24] mentioned that communication, team collaboration, requirements management and verification, languages and cultural differences, temporal and geographical differences, control, scalability, complexity, and coordination are the main challenges of product development. As in GSD, most product development for complex and multi-perspective products consists of different reusable components. Hence, management of components for reusability of components for accurate and appropriate selection difficult process. Subsequently, challenges, trends, and opportunities to manage CBS development are highlighted and described by some existing studies [2,4,7–10]. Therefore, in [2] identified that accurate requirement selection improves architecture of CBS, [4] presented different sources components selection for reusability and challenges of mismatch, ambiguity, and incomplete functionality highlighted, and [9] identified challenges relevant to component requirements mismatch issues during CBS development. Similarly, Nikolaychuk et al. [25] proposed a knowledge-based platform for CBD architecture of components management and implementation of the main CBS functionalities. The CBS existing methods and practices highlighted components development and reused challenges in GSD requirements term mismatch during reuse, selection inconsistency, and ambiguities during integrating components of CBS [10,26–28].

For faults detection different techniques proposed to detect maximum faults i.e., [13–15,22] etc. Thus, Aman et al. [14] and Mohd-Shafie et al. [29] presented a prioritization technique for fault detection based on historical information and model-based methods respectively. Similarly, component-based faults detection enhancement different techniques presented in existing literature, as in [30] proposed method for prioritizing test cases for cost-effective optimization using a search-based algorithm and in [31] provided solution optimizing test cases of configurable components using a weighted search algorithm. The [32] provided a solution for reuse components reliability of CBS and in [4,33] provide the solution for validating components of CBS at runtime. These studies and other studies [34–37] elaborate that CBS testing is a complex and costly process and not efficiently detected faults in CBS. Additionally, in literature different artificial intelligent based methods proposed to manage and detect faults using machine learning [25,38,39] and fuzzy approach [15,40,41] but still need to improve component selection and validation during CBS development.

Hence, few studies describe challenges of CBS management and validation in GSD environment. Thus, Ali et al. [5] presented a framework to resolve change in CBS functionality verification in the cloud environment, and [8] described acceptance testing for maximum faults detection of cloud-based CBS. The author of [22] presented a method to analyze modification impact analysis

in-service orientation architecture using test case prioritization for faults detection. The existing studies described that with an increase in software development using distributed environment using CBS development with an increase in benefits, some challenges of CBS management and validation also raised in GSD [42]. The authors [43] identified those faulty components of CBS that affect the whole system and utilized all available distributed resources and proposed an optimistic monitoring method to identify faulty components in a specific virtual machine. The study [8] provided a fault detection method for CBS requirements acceptance testing in a cloud computing environment. In [7], authors access systematically factors that affect CBS development in GSD. The study [44] provided automated classification and prioritization test cases for product line families. These existing studies [5,8,45–49] defined different challenges, i.e., faults detection, and faults localization according to customers' satisfaction.

Table 1: Compassion of literature

Parameters	[24]	[47]	[9]	[48]	[44]	[42]	[49]	[12]
Coordination and control	○○○	●●●	●○○	○○○	○○○	●●●	●○○	●●●
Lack of semantics	●○○	○○○	●●●	○○○	○○○	○○○	○○○	●○○
Ambiguities	●●●	●●●	●●●	●○○	●○○	●●●	●○○	●○○
Inconsistency	●●●	●○○	●●●	●○○	●○○	●○○	●○○	●○○
Components requirements identification	●●●	●○○	●●●	●○○	○○○	●○○	●○○	●○○
Term mismatch	●○○	○○○	●●●	○○○	○○○	○○○	○○○	○○○
Components requirements Prioritization	●○○	○○○	●○○	○○○	●○○	●○○	●○○	●○○
Components requirements selection	○○○	○○○	●●●	○○○	●○○	●○○	●○○	●○○
Components requirements elicitation	●●●	●○○	●○○	○○○	○○○	●○○	●○○	●○○
Components testing	○○○	○○○	○○○	●●●	●●●	○○○	●○○	○○○
Lack of Artificial intelligent method	○○○	○○○	○○○	○○○	○○○	○○○	●○○	○○○
Faults detection	○○○	○○○	○○○	●●●	●●●	○○○	●○○	○○○

Highlighted = ●●● Partially Highlighted = ●○○ Not Highlighted = ○○○

From existing literature, we analyzed that during reused and integration of components in GSD its complex to be identified, selection and validate components of CBS in GSD due to lack of semantic and term mismatch, diverse perspective of multi-stakeholders, lack communication among stakeholders and project teams, inconsistency, ambiguities, and improper validation activities [7,9,19,50,51]. Comparative analysis of all these existing literatures depicted in Tab. 1. If exiting research work mention any of challenges then recognized as highlighted, and if not then used as not highlighted. In case if describe or highlighted these issues different words and contexts then recognized as partially highlighted. Consequently, we concluded for CBS development in GSD following outcomes of existing literature:

- Communication breakdown among components, stakeholders, and development team required to improve for detecting accurate components requirements in GSD.
- Components accurate and correct selection and priority after components requirement elicitation.
- Faults identification methods are required to improve CBS testing using historical information.
- Components requirements required higher user acceptance to validate CBS during GSD.

Therefore, we proposed an approach for enhancing components selection acceptance testing activities of CBS using the fuzzy approach and historical information in GSD.

3 Proposed Approach

In this section, we propose an approach called a fuzzy approach and historical information (FAHI) approach for CBS. The FAHI approach divided into two main phases as depicted in Fig. 1. The first phase is components selection and priority, which is initiated after the changes request occurs in components requirements from stakeholders and components prioritize using fuzzy approach. Then the second phase is the acceptance testing to verify the changes of requirements in CBS.

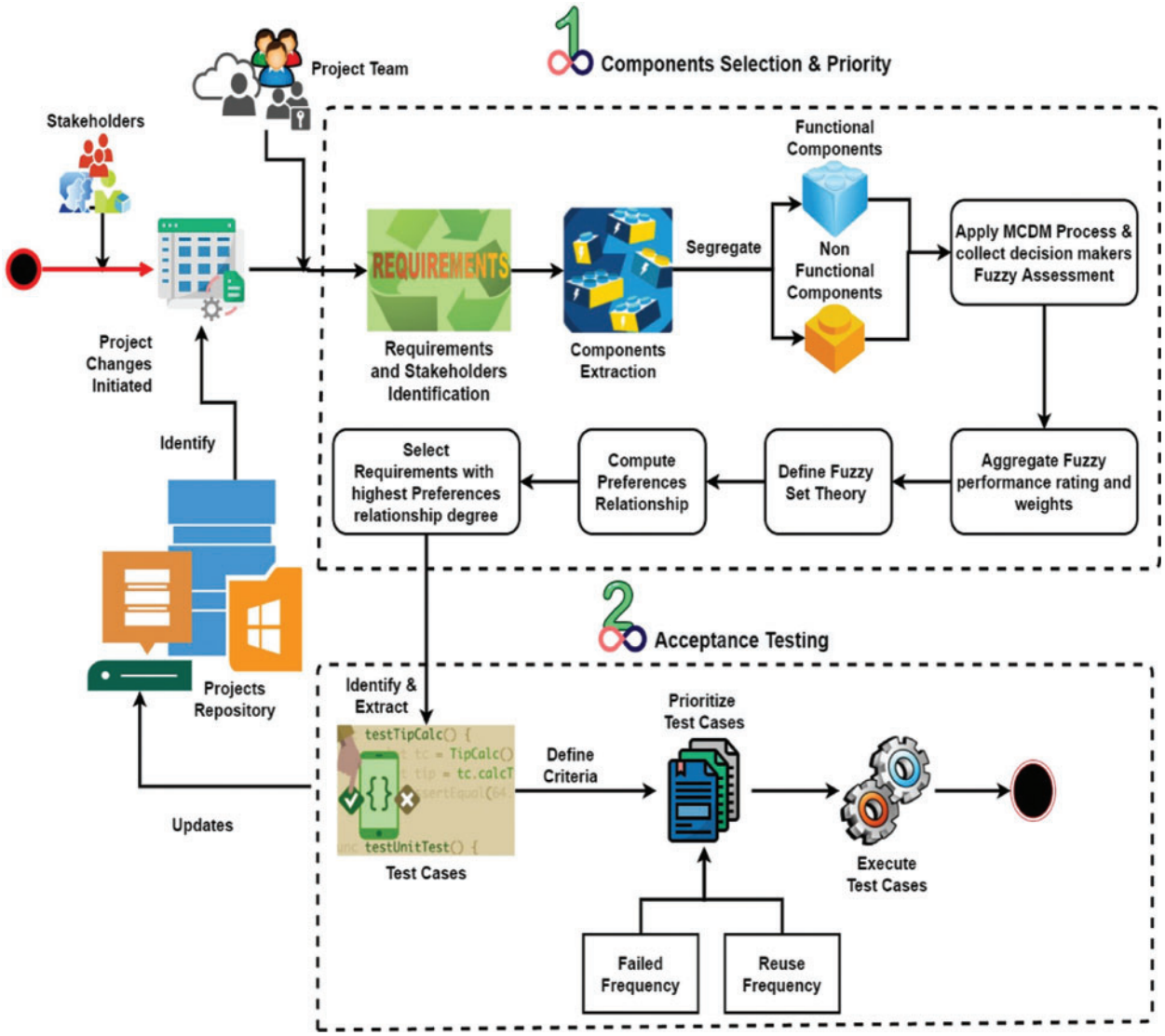

Figure 1: Fuzzy approach and historical information approach

3.1 Components Selection and Priority

The fuzzy approach was adopted to mitigate the challenges of handling changing component requirements and validation of these requirements by accurate selection of components. We have applied the preference relationship on α-level weights to handle multi-criteria decision making for changing components requirements in the pre-development phase. We also have applied linguistic variable set theory in the post-development phase, i.e., handover in GSE. The fuzzy concepts through fuzzy set theory, a linguistic theory of variables, *preferencerelation*(PR), and fuzzy triangular numbers. Set theory is defined as: the universe of discourse $\boldsymbol{U_d}$, a fuzzy set $\boldsymbol{S}$ is characterized by subset function $\boldsymbol{f_S(d)}$,$\boldsymbol{f_S}$: $\boldsymbol{U_d \rightarrow [0,1]}$ and the subset function relates with every subset member of $\boldsymbol{d}$ of $\boldsymbol{U_d}$, where the number of $\boldsymbol{f_S(d)}$ in the interval $\boldsymbol{[0,1]}$ representing subset member ranking of $\boldsymbol{d}$ in $\boldsymbol{S}$.

A linguistic theory of variables is defined as variables with values in form of words or full sentences in natural or machine language [41]. Values of linguistic variable could be in form of responses like; poor, fair, good, average, etc. Fuzzy numbers can be used in the decision-making process and are represented in many forms e.g., Gaussian, Triangular Sigmoid, or Trapezoidal. But most commonly used are triangular fuzzy numbers (TFN) due to ease of use in both conceptual simplicity and complexity [12]. TFN is defined in Eq. (1):

$$\mu_s(d) = \begin{cases} \dfrac{(i-l)}{(j-l)}, & l \le i \le j \\ \dfrac{(k-i)}{(k-j)}, & j \le i \le k \\ 0 & otherwise \end{cases} \tag{1}$$

Let there be a fuzzy universal set comprising all subsets which are represented on a real line "R". Then there exists a piecewise relationship of form, linear function μ_s, where "s" could be convex or normal fuzzy subset members. TFN is then denoted as $S = (l,j,k)$. Preference relation is helpful in the decision-making stage. It is used when group aggregate responses are computed based on individual variable responses. Preference relationship is defined as [16]; PR on R is member subset $\boldsymbol{R^2}$ relationship function $f_{PR}(S, X \text{ } \text{U}S, X \in R2)$, here $f_{PR}(S,X)$represent the range of preference of $\boldsymbol{S}$ over $\boldsymbol{X}$.

(1) PR is reciprocal if, $f_{PR}(S,X) = 1 - f_{PR}(X,S), S, X \in R2$
(2) PR is transitive if, $f_{PR}(S,X) \geq 0.5$ and $f_{PR}(X,Y) \geq 0.5 \rightarrow f_{PR}(S,Y) \geq 0.5 \text{ } \text{U}S, X, Y \in R^2$.

In this phase after requirement elicitation project team identify the requirement of components and stakeholders of the CBS project. Then updated in the team server foundation repository (TFS). In TFS all the stakeholders and project team members communicated and coordinated one to one, synced activities, and monitor projects at every stage. Stakeholders are primary and secondary, primary stakeholders are the ones having direct concern, e.g., decision-makers, financer, beneficiary, etc., while secondary stakeholders are the testers, developers and operators, etc. requirement elicitor is responsible for gathering requirements and objective from primary stakeholders. Identified requirements are segregated into functional and non-functional requirements and constructed based on AND/OR graph. If associated non-functional and sub-requirements are fulfilled then it is AND decomposition and if non-functional is not dependent on any functional requirement then it is OR decomposition. Let the decomposed requirements be

$R1, R2, R3 \ldots Rm$. After this extract components and segregate these components into functional and non-functional components. Then applied the following steps:

3.1.1 Apply MCDM Process and Collect Decision Makers Fuzzy Assessment

Let "N" number of decision-makers (DM) associated with requirement phase (pre-development) for extracting requirements from the requirement gathering stage. Fuzzy assessment of decision-makers is collected according to the assigned importance to each requirement. The importance is given in form of linguistic variable theory. As knowledge and roles differ therefore views also differ.

3.1.2 Aggregate Fuzzy Performance Rating and Weights

Performance rating is done according to weights and the performance matrix is formed based on decision-makers' responses. Scalar multiplication and extended addition are applied to generate the matrix in which performance fuzzy ratings are stored. Performance Fuzzy Rating using Eq. (2) and TFN is formed using $P_{ab} = \left(\frac{1}{n}\right) \odot P^{\frac{1}{ab}} \oplus \sum_{P^{\frac{1}{ab}}}^{P^{\frac{n}{ab}}}$.

Based on which weights are calculated to form the comprehensive vector $\boldsymbol{V}$. Decision matrix ($\boldsymbol{D_m}$) stores comprehensive weights using $D_{m(ab)} = P_{ab} \odot P_b$. This is a fuzzy subset of requirements of the form [15] as shown in Eq. (2):

$$\lambda_{1ab}, \lambda_{2ab}, \lambda_{3ab}, D_{m(ab)}, \Delta_{1ab}, \Delta_{2ab}, \Delta_{3ab} \tag{2}$$

where,

$$\lambda_{1ab} = (V_{2b} - V_{1b})(P_{2ab} - P_{1ab}), \lambda_{2ab} = V_{1b}(P_{2ab} - P_{1ab}) + P_{1ab}(V_{2b} - V_{1b}), \lambda_{3ab} = V_{1b}P_{1ab},$$

$$\Delta_{1ab} = (V_{3b} - V_{2b})(P_{3ab} - P_{2ab}), \Delta_{2ab} = V_{3ab}(P_{3ab} - P_{2ab}) + P_{3ab}(V_{3b} - V_{2b}), \Delta_{3ab} = V_{3ab}P_{3ab}$$

$$D_{ab} = V_{2b}P_{2ab}$$

3.1.3 Define Set Theory for Both Functional and Non-Functional Requirements:

Each requirement is handled as a fuzzy variable number $\boldsymbol{S_a, a = 1, 2, 3, \ldots\ldots n}$ by pre-defined criteria $\boldsymbol{C}$ as shown in Eq. (3).

$$S_a = \frac{1}{C} \odot (D_{m(a1)} \oplus D_{m(a2)} \oplus \cdots \oplus D_{m(aC)}) \tag{3}$$

Parabolic function is defined as follows in Eq. (4) for the above criteria

$$\lambda_{1a}, \lambda_{2a}, \lambda_{3a}, \mathbb{S}_a, \Delta_{1a}, \Delta_{2a}, \Delta_{3a} \tag{4}$$

where,

$$\lambda_{ia} = \frac{1}{C}\sum_{b=1}^{C} \lambda_{i(ab)}, i = 1, 2, 3 \ldots, \quad \Delta_{ia} = \frac{1}{C}\sum_{b=1}^{C} \lambda_{i(ab)}, i = 1, 2, 3 \ldots, \quad \mathbb{S}_a = \frac{1}{C}\sum_{b=1}^{C} D_{m(ab)}$$

3.1.4 Calculate Preference Relationship (PR)

Let preference relationship be $\boldsymbol{PR}$

if $(\lambda_{3a} - \Delta_3) \leq 0, (\Delta_{3a} - \lambda_3) \geq \mathbb{S}_a \geq S, \quad$ *then* $PR_a = PR_{a(1)}$

else if $(\lambda_{3a} - \Delta_3) \leq 0, (\Delta_{3a} - \lambda_3) > 0, \quad \mathbb{S}_a \leq S \quad$ *then* $PR_a = PR_{b(0)}$

else if $(\lambda_{3a} - \Delta_3) = 0, (\Delta_{3a} - \lambda_3) = 0, \quad \mathbb{S}_a = S \quad then \; PR_a = PR_{b(0.5)}$

3.1.5 Select Requirement with Highest Preference Relationship Values

Arrange all the preference relationship values from the decision matrix and sort them in ascending or descending order. Choose the maximum values and shift values as the best alternative.

3.2 Acceptance Testing

In this phase, we extracted test cases of high priority selected components for verifying changes in CBS. The define priority criteria for test case execution to identify maximum faults as soon as possible without irrelevancy and redundancy. For priority of test cases (TC) acceptance, we defined two criteria i.e., frequently reuse and frequently failed. As literature highlighted that multi-criteria improve faults rate and helpful in breaking ties among similar test frequencies. After sorting TC, executing high priority TC to identify maximum faults as earlier as possible. F-measure [5,52] used to verify that selected components' selection accuracy and after bug identification verify the correctness of faults identification rate. For computing faults identification rate used average percentage of faults detection (APFD) [5,15,52].

4 Results and Discussion

In this section, we defined the steps of two experiments performed for FAHI evaluation and described experiment results to investigate and compared FAHI approach effectiveness. The experiment has to approve that FAHI increases faults detection rate with accurate selection and prioritization of components using fuzzy approach during acceptance testing of CBS requirements. Therefore, the complete experiment design is described in Fig. 2. We design two experiments and both experiments consist of pre and post-test on four different CBS projects.

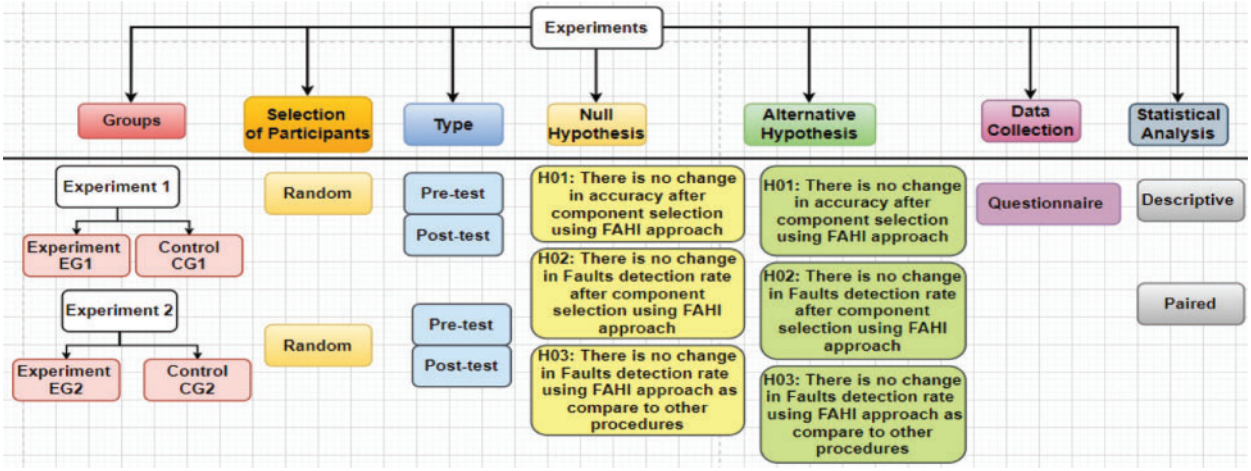

Figure 2: Design overview

The pre-test of both experiments evaluated the current knowledge of participants for CBS development. While in post-test steps after providing training of FAHI approach to EG1 and EG2 and enhanced existing knowledge of CG1 and CG2 to evaluate the effectiveness of FAHI approach. For the effectiveness of the FAHI approach, we compared it with conventional

procedures (CP) i.e., requirement coverage criteria (RCC), and communication coverage criteria (CCC). The FAHI approach implemented for component selection using fuzzy logic and TC priority for acceptance testing used historical information. RCC is used components selection and TC priorities based on highest requirements coverage which means those components which cover maximum requirements functionality selected for validation and used their TC for error identification. Subsequently, CCC used components interaction or communication with each other, which means if one component communicates with maximum components then have high chances of selection and error detection. The CP identifies and validates changes with requirements and code coverage criteria for increasing faults detection rate (FDR). Therefore, we select four real-world projects as dataset and 64 participants of software technologies company as described in Tab. 2. These projects are CBS and consist of different reusable components, historical information, and versions, which are required for experiment conduction. The selected projects save in the organization repository and repository linked to all offices at distributed locations.

Table 2: Projects detail

Detail	Project 1	Project 2	Project 3	Project 4
Constraints	6	8	7	4
Features	35	54	32	18
Requirements	52	68	39	37
Test cases	1110	2450	1450	940
Faults	21	35	18	12
Versions	6	13	4	3

Project 1 (P1), P2, P3, and P4 (i.e., Car alarm, biometric system, health care system, and online management system respectively) are the CBS applications and the project team has been located at different sites. The project team consists of Stakeholders (St), project manager (PM), team leaders (TL), requirement engineer (RE), components designer (Cd), developers (Ds), and quality engineer (QE). The data collection based on a questionnaire and consists of different questions based on some parameters identified from literature, i.e., component selection improves, easy to adopt, coordination and control improves, Communication breakdown among stakeholders improves, the accuracy of component selection and prioritization increases, faults detection increase, historical information helps to increase faults identification, user requirements important for acceptance testing and fuzzy approach important during component selection and prioritization. We implemented the FAHI approach and other procedures for comparison in a software house. The employees of the organization were agreed to implement these procedures for CBS development in GSD to investigate user satisfaction level (SL) and quality of product with accurate component selection and acceptance testing. Therefore, for experiment conduction, we divided participants into an experiment or treatment group (EG) to implement FAHI and a control group (CG) to implement other procedures viz RCC and CCC. The participants in EG is 30 and in CG 34 performed both experiments. In the first experiment we validate P1 and P2, these are desktop applications while in the second experiment we changed the type of project and include web-based CBS, i.e., P3 and P4. The change in type and size of projects helped that FAHI approach useful in different scenarios. The experiments started after the initiation of changes and components selection process conducted with all procedures viz: FAHI approach, RCC, and CCC.

The APFD was used to verify that FAHI (i.e., lies between 90 percent to 100 percent) approach identified faults earlies as compared to RCC and CCC (i.e., lies between 80 percent to 90 percent respectively). This means that the FAHI approach has identified faults as earlier than RCC and CCC procedures. Similarly, in Tab. 3 we compared APFD value with existing research APFD values to define the effectiveness and efficiency of the proposed FAHI approach. Therefore, in Tab. 3 we explained the comparison of APFD and analysis proved that the FAHI approach has a maximum fault detection rate than existing studies.

Table 3: APFD values comparison

Approaches	FAHI	[13]	[15]
APFD	92.80	79.44	75.00

Hereafter, we verified the accuracy of selected components using F-measure. F-measure is a combination of precision and recall to measure the accuracy of procedures effectiveness. The results of F-measures for all procedures are depicted in Fig. 3. The x-axis and y-axis depicted the procedures results and their values respectively. Hence, Fig. 3 results depicted that FAHI approach values more than 85 percent and other procedures both RCC and CCC values less than 85 percent. After components selection accuracy, we performed further steps of experiments to select and prioritize TC. Similarly, Fig. 3 depicted results of all approaches accuracy values of priorities of TC for faults identification accuracy. The x-axis depicted the procedures results against each project and the y-axis describes the priority accuracy values of all procedures.

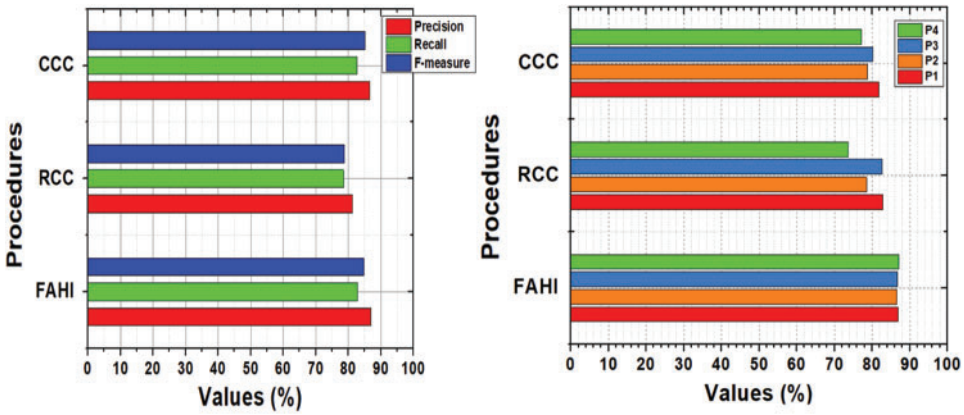

Figure 3: F-measure for components selection and F-measure of TC

For data collection, we used questioner and involved all participants in fulfilling the questionnaire. And results of all project team participants SL of Experiment 1 depicted in Fig. 4, which is more than 65 percent in the case of the FAHI approach and less than 65 percent in other cases like CCC and RCC. Similarly, SL of participants after Experiment 2 more than 70 percent using the FAHI approach and less than 50 percent in RCC and CCC cases as depicted in Fig. 4.

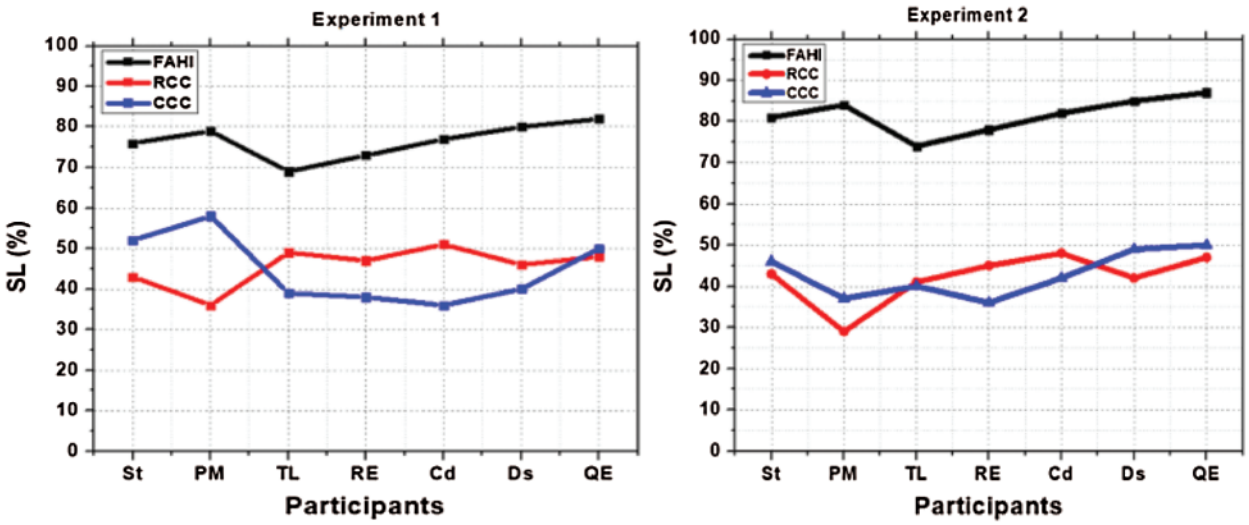

Figure 4: Experiments 1 and 2 SL

Subsequently, to validate questionnaire results used statistical analysis using SPSS 23 software. For statistical analysis, we used two tests i.e., descriptive and hypothesis tests. These tests analysis used for validating procedures effectiveness using three null hypotheses as described in Fig. 2. The results of hypothesis and descriptive testing are described in Tab. 4. The results depicted that there is a significant difference between mean and standard deviation (SD) of all procedures while implementing changes in all projects. The *p*-value, i.e., 0.00 is less than the significance level i.e., 0.05, thus we rejected null hypotheses and alternative hypotheses. Additionally, for evaluating and verifying the FAHI approach for a maximum number of faults identification as compared to RCC and CCC procedures. The FDR results of both experiments for all projects are depicted in Figs. 5 and 6. The x-axis of all projects describes the FDR percentage and y-axis TC execution.

Table 4: Statistical analysis of experiments

Procedure	Projects	Mean	SD	*P*-value	Mean	SD	*P*-value
		Pre-test			Post-test		
FAHI	1	52.80	0.238	0.000	82.80	0.383	0.000
	2	51.80	0.271	0.000	81.80	0.378	0.000
	3	55.40	0.279	0.000	91.40	0.392	0.000
	4	49.80	0.167	0.000	89.80	0.383	0.000
RCC	1	45.40	0.131	0.001	64.30	0.267	0.000
	2	48.60	0.167	0.000	68.40	0.273	0.000
	3	47.00	0.165	0.001	71.00	0.285	0.000
	4	46.50	0.164	0.000	69.50	0.279	0.000
CCC	1	38.91	0.130	0.000	58.19	0.232	0.000
	2	35.29	0.134	0.000	63.91	0.235	0.000
	3	44.45	0.142	0.000	54.15	0.231	0.000
	4	48.51	0.138	0.001	60.51	0.241	0.000

The first highest priority TC of FAHI approach executed and identified more than 90 percent faults during P1 execution as depicted in Fig. 5. While the RCC approach identified 40 percent faults in the first TC execution and CCC identified 20 percent faults in the first TC execution. Therefore, as compared to the second and thirds execution of TC with the FAHI approach there is less need to execute other test cases because maximum faults are identified without any ambiguities, irrelevancy, and redundancy. If these faults are removed earlier then testing effort reduces and user requirements acceptance increases with higher product quality and utilization of fewer resources. Subsequently, in other cases the more utilization of development resources and less productivity due to irrelevant test selection, redundant test priority, and redundant faults detection. As faults identification not stable and changed in every execution of RCC and CCC TC. This reduces FDR and increases faults redundancy.

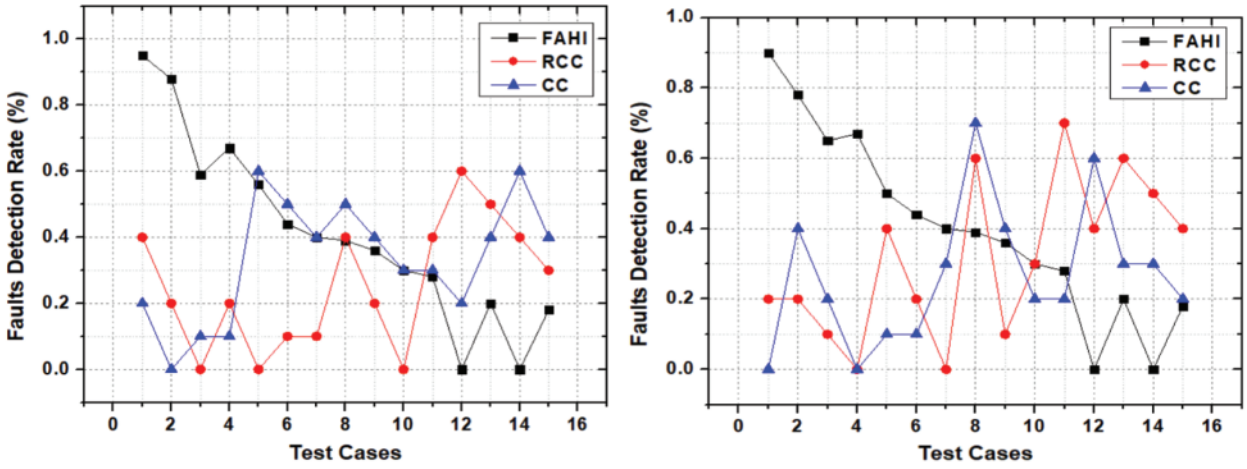

Figure 5: FDR for P1 and P2

For example, if 2 faults out of 5 identified in the first execution of TC and 3 identified in the second TC execution. These three faults may have only identified 1 new fault and two old faults which were identified during the first TC execution. This results in redundant faults and increases efforts with less productivity. Therefore, it proves that FAHI significantly performed during the development of different CBS types of projects irrespective of size, type, and development location without any coordination, time zone, and limited resources issues. Hence, in other P2, P3, and P4 similar results were identified which conclude while describing results of P1. It means that the FAHI approach identifies in all projects after first execution more than 90 percent faults without any redundancy or repetitions in faults and decrease while executing TC further. Similarly, RCC and CCC identified less than 50 percent faults while first TC execution and get similar faults in further TC execution. Which increase time and efforts of the project team and ultimately impact software quality due to a decrease in user acceptance rate as depicted in Figs. 5 and 6.

At last, all results concluded that FAHI approach resolves components selection and validation after changes to increase user satisfaction and requirements in the global software development environment. And FAHI approach identified maximum faults as earlier as possible in comparison to RCC and CCC procedures. The FAHI approach outperform from existing practices, i.e., [13,15,22,30,31,34] in terms of improving faults detection rate using fuzzy approach.

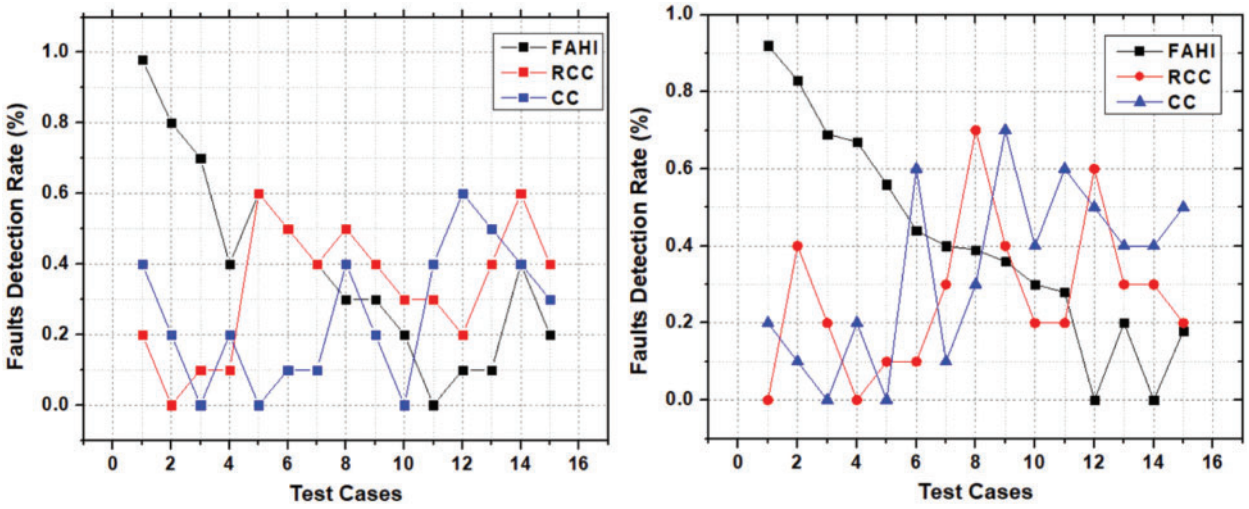

Figure 6: FDR for P3 and P4

Therefore, after changes in component functionality, there are chances of term mismatch during the selection of the specifications of multi-perspective stakeholder components during function and non-functional components functionality. As most of the changes were initiated from the market and stakeholders' reviews and feedback in natural language. Therefore, for actual and sematic based component functionality extraction required aspect based sentimental analysis to improve validity and increase faults detection of CBS after changes.

Hence, for experimental assessment, numerous threats arise disagreement among theoretical and practical rationality of literature and experimental results. Thus, it requires repetition in research to accept or refute findings. The basic main four threats are: internal validity (IV), external validity (EV), construct validity (CV), and reliability validity (RV) [13,52,53].

IV is related to factor identification and critical analysis regarding components selection and testing. To report IV threat, different resources to search relevant existing industrial and practical studies to develop a theory for critical analysis of studies and challenges of CBS in GSD. And another threat relevant to IV is the unbiased and rationality of data collection during the empirical evaluation of FAHI we used a questionnaire and statistical analysis for validation of the questionnaire. EV concerns to FAHI practical applicability and outcomes in industrial projects. Therefore, we used four industrial projects of different sizes and functionality in distributed organizations for FAHI approach applicability and compared results with conventional procedures for generalization analysis.

The relations between the different concepts and reflections are considered by CV. This includes the use of various metrics to determine the validity of the various component selection activities and acceptance testing criteria using FAHI as compared to other procedures. After introducing FAHI and other procedures, the CV threat relevant to the identification of accurate satisfaction level of participants. Therefore, we mitigate this threat we used questioner based on factors identified from the literature and used multi-project for evaluation using two different pre and post-test experiments. Later, we compared statistical results of pre and post-tests observation for reducing conflicts and provide guidelines to researchers regarding this research work validity. RV relates to the associations between action and consequence. This can be mitigated via a

rigorous practical assessment of the various decisions employed for FAHI authentication. Thus, to mitigate RV threat all authors participated in evaluation for data collection and counteractions on analysis of results with a multi-perspective of stakeholders and project team.

5 Conclusion and Future Work

In this research work, we proposed a FAHI approach to enhance the selection of components and acceptance testing after requirement change for component-based product production in GSD. This reduces component selection, higher user acceptance, and project team coordination in GSD using fuzzy logic and historical information. The fuzzy logic method helped in the identification of high priority components for selection and historical information, i.e., frequently reuse and change components information helped to prioritized test cases. Thus, fuzzy logic also reduces irrelevancy during the selection of components and test cases while historical information helped in reducing redundancy in test cases and identified faults. For effectiveness analysis of the proposed FAHI approach, an experimental study was conducted with statistical. The results show that the proposed FAHI approach has a significant mean difference and higher satisfaction, i.e., greater than 80 percent as compared to other procedures viz: RCC and CCC, i.e., less than 80 percent in CBS. Thus, the proposed FAHI approach enhanced the selection of CBS components during product development and increase detection of faults rate distributed environment. Therefore, the FAHI approach provides a roadmap for researchers and industrialists in the domain of CBS user acceptance testing activities. In future work, we will be extended our proposed framework in a cloud computing environment to resolve specification, prioritization, and testing issues of CBS. And mitigate the problem of quality analysis in the embedded components development environment after continuous modification of CBS.

Funding Statement: Taif University Researchers Supporting Project No. (TURSP-2020/10), Taif University, Taif, Saudi Arabia.

Conflicts of Interest: There are no conflicts of interest to report regarding the present study.

References

[1] I. Gambo, R. Ikono, P. Achimugu and A. Soriyan, "An integrated framework for prioritizing software specifications in requirements engineering," *International Journal of Software Engineering and Its Applications*, vol. 12, no. 1, pp. 33–46, 2018.

[2] B. Imam Ya'u and M. Nura Yusuf, "Building software component architecture directly from user requirements," *International Journal of Engineering and Computer Science*, vol. 7, no. 2, pp. 23557–23566, 2018.

[3] V. Singhal, S. S. Jain, D. Anand, A. Singh, S. Verma *et al.,* "Artificial intelligence enabled road vehicle-train collision risk assessment framework for unmanned railway level crossings," *IEEE Access*, vol. 8, pp. 113790–113806, 2020.

[4] M. Borg, P. Chatzipetrou, K. Wnuk, E. Alégroth, T. Gorschek *et al.,* "Selecting component sourcing options: A survey of software engineering's broader make-or-buy decisions," *Information and Software Technology*, vol. 112, no. 8, pp. 18–34, 2019.

[5] S. Ali, Y. Hafeez, N. Z. Jhanjhi, M. Humayun, M. Imran *et al.,* "Towards pattern-based change verification framework for cloud-enabled healthcare component-based," *IEEE Access*, vol. 8, pp. 148007–148020, 2020.

[6] S. Y. Shin, S. Nejati, M. Sabetzadeh, L. C. Briand and F. Zimmer, "Test case prioritization for acceptance testing of cyber physical systems: A multi-objective search-based approach," in *Proc. of the 27th ACM SIGSOFT Int. Sym. on Software Testing and Analysis*, Amsterdam, Netherlands, pp. 49–60, 2018.

[7] M. J. Khan and S. Mahmood, "Assessing the determinants of adopting component-based development in a global context: A client-vendor analysis," *IEEE Access*, vol. 6, pp. 79060–79073, 2018.

[8] M. Smara, M. Aliouat, A. S. K. Pathan and Z. Aliouat, "Acceptance test for fault detection in component-based cloud computing and systems," *Future Generation Computer Systems*, vol. 70, no. 1, pp. 74–93, 2017.

[9] C. Ayala, A. Nguyen-Duc, X. Franch, M. Höst, R. Conradi *et al.,* "System requirements-OSS components: Matching and mismatch resolution practices–An empirical study," *Empirical Software Engineering*, vol. 23, no. 6, pp. 3073–3128, 2018.

[10] B. Graics, V. Molnár, A. Vörös, I. Majzik and D. Varró, "Mixed-semantics composition of statecharts for the component-based design of reactive systems," *Software and Systems Modeling*, vol. 19, no. 6, pp. 1483–1517, 2020.

[11] M. Shameem, B. Chandra, C. Kumar and A. A. Khan, "Impact of requirements volatility and flexible management on GSD project success: A study based on the dimensions of requirements volatility," *International Journal of Agile Systems and Management*, vol. 12, no. 3, pp. 199–227, 2019.

[12] 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 Software*, vol. 14, no. 4, pp. 389–401, 2020.

[13] M. A. Hasan, M. A. Rahman and M. S. Siddik, "Test case prioritization based on dissimilarity clustering using historical data analysis," in *Int. Conf. on Information, Communication and Computing Technology*, Singapore, Springer, pp. 269–281, 2017.

[14] H. Aman, T. Nakano, H. Ogasawara and M. Kawahara, "A topic model and test history-based test case recommendation method for regression testing," in *Int. Conf. on Software Testing, Verification and Validation Workshops*, Vasteras, pp. 392–397, 2018.

[15] C. Hettiarachchi and H. Do, "A systematic requirements and risks-based test case prioritization using a fuzzy expert system," in *Proc. 19th IEEE Int. Conf. on Software Quality, Reliability and Security*, Sofia, Bulgaria, pp. 374–385, 2019.

[16] 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," *Applied Soft Computing*, vol. 90, no. 5, 106122, 2020.

[17] R. Sinha, M. Shameem and C. Kumar, "SWOT: Strength, weaknesses, opportunities, and threats for scaling agile methods in global software development," in *Proc. of the 13th Innovations in Software Engineering Conf. on Formerly Known as India Software Engineering Conf.*, Jabalpur India, pp. 1–10, 2020.

[18] A. A. Khan, J. Keung, M. Niazi, S. Hussain and M. Shameem, "GSEPIM: A roadmap for software process assessment and improvement in the domain of global software development," *Journal of software: Evolution and Process*, vol. 31, no. 1, e1988, 2019.

[19] J. M. Carrillo de Gea, J. D. Nicolás, J. L. Fernández-Alemán and A. Toval, "Automated support for reuse-based requirements engineering in global software engineering: Automated support for reuse-based RE in global software engineering," *Journal of Software: Evolution and Process*, vol. 29, no. 8, e1873, 2017.

[20] A. A. Alsanad, A. Chikh and A. Mirza, "A domain ontology for software requirements change management in global software development environment," *IEEE Access*, vol. 7, pp. 49352–49361, 2019.

[21] T. Kamal, Q. Zhang and M. A. Akbar, "Toward successful agile requirements change management process in global software development: A client-vendor analysis," *IET Software*, vol. 14, no. 3, pp. 265–274, 2020.

[22] H. Wang, M. Yang, L. Jiang, J. Xing, Q. Yang *et al.,* "Test case prioritization for service-oriented workflow applications: A perspective of modification impact analysis," *IEEE Access*, vol. 8, pp. 101260–101273, 2020.

[23] E. H. Trainer and D. F. Redmiles, "Bridging the gap between awareness and trust in globally distributed software teams," *Journal of Systems and Software*, vol. 144, no. 10, pp. 328–341, 2018.

[24] C. Burnay, S. Bouraga, J. Gillain and I. J. Jureta, "What lies behind requirements? A quality assessment of statement grounds in requirements elicitation," *Software Quality Journal*, vol. 8, pp. 1–29, 2020.

[25] O. A. Nikolaychuk, A. I. Pavlov and A. B. Stolbov, "The software platform architecture for the component-oriented development of knowledge-based systems," in *2018 41st Int. Convention on Information and Communication Technology, Electronics and Microelectronics*, Opatija, pp. 1064–1069, 2018.

[26] M. Arias, A. Buccella and A. Cechich, "A framework for managing requirements of software product lines," *Electronic Notes in Theoretical Computer Science*, vol. 339, pp. 5–20, 2018.

[27] M. Chakraborty and N. Chaki, "A new framework for configuration management and compliance checking for component-based software development," in *Advanced Computing and Systems for Security*. New Delhi: Springer, pp. 173–188, 2016.

[28] V. D. Moreno, G. Génova, E. Parra and A. Fraga, "Application of machine learning techniques to the flexible assessment and improvement of requirements quality," *Software Quality Journal*, vol. 56, no. 3, pp. 1–30, 2020.

[29] M. L. Mohd-Shafie, W. M. N. Wan-Kadir, M. Khatibsyarbini and M. A. Isa, "Model-based test case prioritization using selective and even-spread count-based methods with scrutinized ordering criterion," *PLoS One*, vol. 15, no. 2, e0229312, 2020.

[30] A. Arrieta, S. Wang, G. Sagardui and L. Etxeberria, "Search-based test case prioritization for simulation-based testing of cyber-physical system product lines," *Journal of Systems and Software*, vol. 149, no. 3, pp. 1–34, 2019.

[31] S. Y. Shin, K. Chaouch, S. Nejati, M. Sabetzadeh, L. C. Briand *et al.,* "Uncertainty-aware specification and analysis for hardware-in-the-loop testing of cyber-physical systems," *Journal of Systems and Software*, vol. 171, 110813, 2020.

[32] A. Mohan and S. K. Jha, "Predicting and accessing reliability of components in component based software development," in *2019 Int. Conf. on Intelligent Computing and Control Systems*, Madurai, India, pp. 1110–1114, 2019.

[33] S. Shirata, H. Oyama and T. Azumi, "Runtime component information on embedded component systems," in *2018 IEEE 16th Int. Conf. on Embedded and Ubiquitous Computing*, Bucharest, pp. 166–173, 2018.

[34] G. Buchgeher, S. Fischer, M. Moser and J. Pichler, "An early investigation of unit testing practices of component-based software systems," in *2020 IEEE Workshop on Validation, Analysis and Evolution of Software Tests*, London, ON, Canada, pp. 12–15, 2020.

[35] M. Azizi and H. Do, "A collaborative filtering recommender system for test case prioritization in web applications," in *Proc. of the 33rd Annual ACM Sym. on Applied Computing*, New York, NY, USA: Association for Computing Machinery, pp. 1560–1567, 2018.

[36] M. Al-Hajjaji, T. Thüm, M. Lochau, J. Meinicke and G. Saake, "Effective product-line testing using similarity-based product prioritization," *Software & Systems Modeling*, vol. 18, no. 1, pp. 499–521, 2019.

[37] R. He, L. Tang, X. Han, W. He and Z. Zhao, "Software component reliability evaluation method based on characteristic parameters," in *2018 IEEE Int. Conf. on Software Quality, Reliability and Security Companion*, Lisbon, pp. 235–241, 2018.

[38] A. J. Fernández-García, L. Iribarne, A. Corral, J. Criado and J. Z. Wang, "A recommender system for component-based applications using machine learning techniques," *Knowledge-Based Systems*, vol. 164, no. 1, pp. 68–84, 2019.

[39] K. Sheoran, P. Tomar and R. Mishra, "A novel quality prediction model for component-based software system using ACO-NM optimized extreme learning machine," *Cognitive Neurodynamics*, vol. 14, no. 4, pp. 509–522, 2020.

[40] M. G. Akbari and G. Hesamian, "Testing statistical hypotheses for intuitionistic fuzzy data," *Soft Computing*, vol. 23, no. 20, pp. 10385–10392, 2019.

[41] Y. Wang, F. Subhan, S. Shamshirband, M. Z. Asghar, I. Ullah *et al.,* "Fuzzy-based sentiment analysis system for analyzing student feedback and satisfaction," *Computers, Materials & Continua*, vol. 62, no. 2, pp. 631–655, 2020.

[42] M. A. Akbar, S. Mahmood, H. AlSalman, A. Razzaq, A. Gumaei *et al.,* "Identification and prioritization of cloud based global software development best practices," *IEEE Access*, vol. 8, pp. 191242–191262, 2020.

[43] I. Gonzalez-Herrera, J. Bourcier, E. Daubert, W. Rudametkin, O. Barais *et al.,* "ScapeGoat: Spotting abnormal resource usage in component-based reconfigurable software systems," *Journal of Systems and Software*, vol. 122, no. 12, pp. 398–415, 2016.

[44] I. Hajri, A. Goknil, F. Pastore and L. C. Briand, "Automating system test case classification and prioritization for use case-driven testing in product lines," *Empirical Software Engineering*, vol. 25, no. 5, pp. 3711–3769, 2020.

[45] C. Yang, J. Liu, Y. Zeng and G. Xie, "Real-time condition monitoring and fault detection of components based on machine-learning reconstruction model," *Renewable Energy*, vol. 133, no. 4, pp. 433–441, 2019.

[46] X. Li, W. E. Wong, R. Gao, L. Hu and S. Hosono, "Genetic algorithm-based test generation for software product line with the integration of fault localization techniques," *Empirical Software Engineering*, vol. 23, no. 1, pp. 1–51, 2018.

[47] G. Borrego, A. L. Morán, R. R. Palacio, A. Vizcaíno and F. O. García, "Towards a reduction in architectural knowledge vaporization during agile global software development," *Information and Software Technology*, vol. 112, no. 8, pp. 68–82, 2019.

[48] F. Dadeau, J. P. Gros and O. Kouchnarenko, "Testing adaptation policies for software components," *Software Quality Journal*, vol. 28, no. 3, pp. 1347–1378, 2020.

[49] M. Oriol, S. Martínez-Fernández, W. Behutiye, C. Farré, R. Kozik *et al.,* "Data–driven and tool-supported elicitation of quality requirements in agile companies," *Software Quality Journal*, vol. 28, no. 3, pp. 931–963, 2020.

[50] A. Ibias, R. M. Hierons and M. Núñez, "Using squeeziness to test component-based systems defined as finite state machines," *Information and Software Technology*, vol. 112, no. 8, pp. 132–147, 2019.

[51] T. Lu, C. Liu, H. Duan and Q. Zeng, "Mining component-based software behavioral models using dynamic analysis," *IEEE Access*, vol. 8, pp. 68883–68894, 2020.

[52] S. Ali, Y. Hafeez, S. Hussain and S. Yang, "Enhanced regression testing technique for agile software development and continuous integration strategies," *Software Quality Journal*, vol. 28, no. 2, pp. 397–423, 2020.

[53] E. Bjarnason, H. Sharp and B. Regnell, "Improving requirements-test alignment by prescribing practices that mitigate communication gaps," *Empirical Software Engineering*, vol. 24, no. 4, pp. 2364–2409, 2019.