Computers, Materials & Continua

Tech Science Press

DOI: 10.32604/cmc.2024.051371

ARTICLE

Enhancing Secure Development in Globally Distributed Software Product Lines: A Machine Learning-Powered Framework for Cyber-Resilient Ecosystems

Marya Iqbal[1], Yaser Hafeez[1], Nabil Almashfi[2], Amjad Alsirhani[3], Faeiz Alserhani[4], Sadia Ali[1], Mamoona Humayun[5,*] and Muhammad Jamal[6]

[1]University Institute of Information Technology, PMAS-Arid Agriculture University, Rawalpindi, Pakistan

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

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

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

[5]Department of Information Systems, College of Computer and Information Sciences, Jouf University, Sakaka, Al Jouf, 72388, Saudi Arabia

[6]Department of Mathematics, Pir Mehr Ali Shah Arid Agriculture University, Rawalpindi, Pakistan

*Corresponding Author: Mamoona Humayun. Email: mahumayun@ju.edu.sa

Received: 04 March 2024 Accepted: 06 May 2024 Published: 20 June 2024

ABSTRACT

Embracing software product lines (SPLs) is pivotal in the dynamic landscape of contemporary software development. However, the flexibility and global distribution inherent in modern systems pose significant challenges to managing SPL variability, underscoring the critical importance of robust cybersecurity measures. This paper advocates for leveraging machine learning (ML) to address variability management issues and fortify the security of SPL. In the context of the broader special issue theme on innovative cybersecurity approaches, our proposed ML-based framework offers an interdisciplinary perspective, blending insights from computing, social sciences, and business. Specifically, it employs ML for demand analysis, dynamic feature extraction, and enhanced feature selection in distributed settings, contributing to cyber-resilient ecosystems. Our experiments demonstrate the framework's superiority, emphasizing its potential to boost productivity and security in SPLs. As digital threats evolve, this research catalyzes interdisciplinary collaborations, aligning with the special issue's goal of breaking down academic barriers to strengthen digital ecosystems against sophisticated attacks while upholding ethics, privacy, and human values.

KEYWORDS

Machine Learning; variability management; cybersecurity; digital ecosystems; cyber-resilience

1 Introduction

The purpose of software engineering is to create quality software within budget and on time [1–4]. Application engineering (AE) involves the construction of individual software systems. The reuse of software resources in AE saves developers time and effort [3,5,6]. However, this ad-hoc process presents numerous issues. Over time, software complexity increases, necessitating changes to the development process to maintain high quality. In the contemporary software industry, the top priority for software developers is to produce high-quality software. Agile-based development has replaced conventional development methodologies in the software industry [3,7,8].

Organizations are increasingly adopting agile approaches in the fast-paced software development landscape to deliver high-quality software products quickly and flexibly. Agile development practices place a premium on collaboration, iterative development, and continuous improvement, allowing teams to adjust swiftly to changing requirements and client expectations. Teams that collaborate need to coordinate. As a result, Scrum masters collaborate with each team to plan activities, handle dependencies, and prevent effort duplication [5,9]. Recent trends in software engineering involve working with geographically dispersed teams. This globalization of software development arises from clear business goals, including addressing the lack of local IT skills, reducing development costs, and supporting offshoring and outsourcing. These factors will be stronger later, and we confront more software development globalization [4].

To handle these problems global software engineering (GSE) was introduced. In software development, GSE represents a new paradigm as a coordinated activity, dispersed geographically rather than centralized. GSE literature shows that its primary concern is to develop a single system [10,11]. However, they often aimed to develop the product line from many organizations [10]. Scrum is a well-known agile framework that has been widely used in software development because of its collaborative and iterative methodology. The practice of reusing parts of existing software to create new software, rather than starting from scratch, is known as software reuse. However, it may also lead to the development of new creative solutions for challenging technical problems [3,10]. To develop families of systems through reuse, software product lines engineering (SPLE) is an approach [5]. Because of advancement, often changing requirements, and quick release in the software industry, organizations develop a family of products by reusing certain fundamental assets with modifications rather than creating new products and versions [12]. SPLE paradigm has been proved successful in industry [13,14].

The main advantages of the SPLE are reduced time-to-market, reduced cost and improved quality. Therefore increasing the concept of reusability of SPL in Component Based Software Development (CBSD) to reduce time and cost for high quality productivity [15–17]. The SPLE consists of two parts that are domain engineering (by specifying commonalities and variability in product families, it establishes a reusable platform) and application engineering (it takes responsibility of product mining utilizing a plat-form for domain engineering and a mix of product development's related processes, such as requirement engineering and application design) [18,19].

Processes for developing software have lately undergone a radical transformation. There is a growing need for quick development and adaptability due to the increasingly rapid speed of varying requirements on the way to meet market needs as well as to be compliant with evolving norms and standards [1,2,20]. Thus, a contemporary problem that many firms must face is continuous software improvement of software processes. There are many current methods for process improvement. While there are many current methods for process improvement, it is believed that current development procedures are too slow to meet future demands [21]. Agile Software Development (ASD) finds a key to this issue. Better software quality, a quicker feedback cycle, and taking less time to market

every feature is promised [7,21,22]. As a result, ASD is an excellent topic for software process improvement [23].

SPLE must be used in addition to global software development (GSD) in order to enhance product quality and customer collaboration because useful knowledge management and reusability are disregarded in GSD [3]. The dynamic variability (DV) management prevailing in GSD based agile SPL poses special difficulties [24]. A promising method to manage DV in SPL and ensure security in Scrum and GSD teams is machine learning (ML) based Dynamic Variability Management (DVM). Hence, improve performance of development and get rid of security risks by using ML algorithms to configure and automate software analysis, optimization, and decision-making processes.

The purpose of this study is to use ML-DVM to secure development practices and improve DVM for SPL in GSD with ASD. Cyber-resilient ecosystems are dynamic and constantly changing. Traditional sequential and plan-driven approaches to software development may not be well-suited to these environments. Agile development methods, on the other hand, place more emphasis on adaptation, flexibility, and iterative development, which is more appropriate given the quick speed of change involved in building cyber-resilient systems [3,21,25]. Agile paradigms have advantages in the context of ML-DVM, including ability to react quickly to changing requirements, incorporate stakeholder feedback, and provide progressive improvements over time [25–29]. However, issues may develop in integrating Agile's iterative character with the complex and sometimes unpredictable nature of ML algorithms and data. This could result in conflicts between agile principles like defined iteration lengths and naturally non-linear growth of ML model construction. As a result, while Agile approaches offer a promising foundation for improving safe development inside GSD for SPL, careful study and adaptation are required to successfully combine them with particular problems and opportunities presented by ML-DVM. Adopting hybrid Agile approaches and fostering collaboration between development teams and data scientists can ensure alignment between development goals and ML outcomes.

Therefore, addressing challenges such as handling irrelevancy, coordination, task allocation, multiplatform involvement, resource allocation, and the roles of ML in Agile-based GSD for SPL is crucial. Consequently, identified that there is a need for a comprehensive approach to manage DV and secure the development process by providing one platform and features analysis using ML. Thus, we proposed a framework to manage dynamic variability of features and secure development process in GSD with Agile methodologies using ML. Hence, the core contribution of the study is to recognize challenges/barriers of existing SPL techniques especially in the GSD environment using Agile and propose an approach to mitigate these challenges to improve software quality and team collaboration using ML approach [1,24]. Data mining reveals hidden information patterns that aid in the prioritization of requirements. Following classification, each categorized requirement's frequently accessed characteristics were discovered and designated as significant features for the prioritization process [15,21,30].

The contributions of this study are as follows:

- This study examines the fundamental components of ML-DVM, including data collection, analysis, model training, and adaptive decision-making. Also discuss the unique security concerns and difficulties that arise while building SPL with Scrum in an agile, GSD.
- To increase quality and efficiency, a framework for agile SPLE based systems was presented to control variability and relevant component selection based on user feedback using ML to create a feature model for viability management.

- An empirical study is conducted to analyze performance of proposed framework in order to assess its effectiveness. Conducted a comparison analysis to assess usefulness of suggested framework in terms of commonality and variability management in SPL based systems in order to give practitioners and researchers with guidance.
- Provide critical insights and practical guidance to developing teams and researchers so they can establish safe and efficient development practices for SPL.

Section 2 provides review of existing literature and Section 3 proposes solution of the identified problem from related work. Furthermore, Section 4 employs empirical research to evaluate the suggested framework. In Section 5, we conclude the study and discussed some possible directions for future research in SPL software engineering.

2 Related Work

In this section, we discuss issues related to SPL in GSD based on research findings and previously published literature. In the field of software development, there has been a lot of interest in the SPL concept. In order to understand SPL, consider that a product line is a managed group of software products that have certain features in common and can be customized to meet the needs of different markets within a given domain [1]. The advantages of SPLs have been acknowledged by many studies; however, it is important to comprehend the difficulties associated with current frameworks and approaches in this field [2]. While SPLs have the potential to improve product quality and allow agile development practices, they also pose significant challenges that must be overcome. These difficulties have prompted researchers and practitioners to explore deeper into the realm of SPLs in quest of novel methods to overcome existing constraints.

The SPLE research community has made tremendous developments, producing high-quality research and recommending standardized frameworks. This article focuses on the difficulties encountered during scoping, quality assurance, application requirements, domain design, and realization. In the next decade, emphasized trends include resolving the open world supposition, controlling unpredictability in non-product-line contexts, and exploiting instantaneous input [31]. Feature modeling is a vital activity of SPLE, developing requirements models for product families and giving direction for single products. Feature selections satisfy client's requirements and between features, they must obey interrelationships. Once associated features are chosen, they must further propose other reusable assets for implementation, such as components, test cases, etc. The aspect-oriented framework is proposed to analyze the quality-based features of SPL and their interrelationship. However, there is a lack of variability management [11,21,30].

Large-scale transformation models already in existence are identified and analyzed for defining an agile transformation model for large companies preserving SPLs. But the limitation in this paper is large companies can utilize the given model as guidance throughout the agile process of transformation, and this model is more of an organizational approach rather than a development approach so despite this model, an organization's integration framework is still needed [3,10,14]. During the concurrent evolution of various products, SPL may be extracted and maintained using agile and semi-automated methods. However, a process retrieves and maintains SPL without providing any process (information) [3,18,19].

GSD is carried out by teams situated in distinct locations of the globe who grow economically feasible software for an organization. The economic benefits of GSD generate the interest of implementing GSD in the international software industry [21,23]. The volume of documents, models,

software code, and other relevant artifacts has increased significantly due to the exponential growth of GSD efforts. Reuse has gained a lot of favor in recent years. Significant benefits of software reuse include cost savings, resource reduction, and a shorter development cycle. However, it is a difficult undertaking to identify artifacts from a specific enterprise's repositories.

In two large software-producing organizations [18], reuse practices are compared. Both organizations do not take the benefit of other accessible artifacts, only implement pragmatic reuse of code, not leveraging other accessible artifacts. Reusable elements regained from a repository, if found, else, direct communication with trusted colleagues is critical for access. They do not deal with change in design and not focus on usability practices [32]. Effective reuse is closely related to organizational goals and is deeply rooted in development traditions, both in regard to tool management and support. To better plan, comprehend, and control work allocation choices, GSD organizations should aim to take into account the highlighted usability elements while managing their operations of GSD [33]. Therefore, there is a need for an effective and valuable methodology that can handle all above issues and can improve SPL practices using modern technologies in GSD. The overview about the literature summarizes in Table 1.

Table 1: Literature overview

Parameters	[3]	[5]	[7]	[14]	[15]	[19]	[22]	[23]	[21]
Documentation analysis	□□	■■	■■	□□	■■	■■	■■	■■	■□
Variability management	■■	■□	■■	■■	■□	■□	■□	■□	□□
Redundancy	□□	■□	■■	■■	■□	■□	■□	■□	□□
Component management	■□	■□	■■	■■	■■	■□	■■	■■	□□
Communication	■□	□□	■□	■□	■■	□□	□□	□□	■■
Coordination	□□	□□	■□	■□	■■	□□	■□	□□	■■
Control	■■	□□	■□	■□	■■	□□	■□	■□	■■
Changing components requirements	■□	□□	■■	■■	■■	■□	■■	■□	■■
Component selection	□□	■□	■■	□□	■□	□□	■□	■□	■□
Task allocation	□□	■■	■■	■□	■■	■□	■■	■■	□□
Team management	□□	□□	■■	□□	□□	□□	□□	■■	□□
Machine learning	□□	□□	■■	□□	□□	□□	□□	■■	□□
Domain knowledge	□□	□□	■■	□□	□□	□□	□□	■■	□□
Agile methodologies	■■	□□	■□	■■	□□	■■	■□	■□	□□
Specified = ■■	Partially specified = ■□					Not specified = □□			

3 Material and Methods

To address the identified problems, we propose a framework (PF) that provides a single platform instead of multiple platforms by utilizing a common repository called Team Foundation Server (TFS) for knowledge and resource management. This framework aims to increase productivity and user satisfaction within limited and globally distributed software development environments. The proposed framework utilizes the TFS common repository to manage SPL in GSD. The benefit of employing the logical concept of TFS repositories is that each client and team member does not need to install

different types of tools for coordination and communication during software development, covering the entire lifecycle of applications.

Microsoft Visual Studio and Eclipse are integrated into TFS for all platforms, allowing various Integrated Development Environments (IDEs) to utilize it as their backend. These tools function independently in different domains: TFS maintains the repositories and project management, while Microsoft Visual Studio and Eclipse are used for project development and testing rather than repository management. Thus, the framework enhances the performance of the project management team by customizing these tools for project management, repository management, and software testing on a single platform, reducing the need for multiple platforms in a distributed environment to improve communication and coordination among teams, thereby enhancing efficiency and performance.

However, based on literature findings, the validation of selected features is an important aspect during development. Therefore, we integrated requirements and testing for feature extraction and selection dynamically for reuse to manage variability using Machine Learning (ML) during SPL in GSD. The main objective of our framework is to manage variabilities and select the correct features and components based on reuse requirements present in the repository to address new requirements of SPL products in GSD using ML methods. Validation of selected features is another aspect for Dynamic Verification and Validation Methodology (DVM) to validate more accurate and relevant component selections for reuse and their integration into SPL products to enhance overall feature quality systematically.

In our framework, we include the testing process typically used in Scrum, i.e., Agile methodology. This integration ensures accurate requirement analysis for feature creation based on semantic analysis. Classification and selection algorithms are then utilized to find correct relevant features for reuse. Subsequently, relevant test cases are selected for validation based on frequently reused selected features for DVM to improve the reliability and effectiveness of the framework during SPL development in GSD.

The reason for combining ML-DVM is to improve secure development of SPL in Agile-based GSD. This resolves communication and coordination issues by aligning feature development in SPL from requirements to testing, facilitating a unified and effective development process systematically. It ensures the validity of features from selection to testing and also improves collaboration between teams and clients by providing a unified platform. For SPL, the following steps are adopted in the GSD environment. The PF is depicted in Fig. 1.

3.1 Requirement Elicitation

Clients are located at different geographical locations, sites A, B, and C. The project manager gathers or elicits requirements from the dispersed clients at various locations through online interviews, video calls, and using interfaces such as Skype. After elicitation, the project manager creates user stories of requirements and removes conflicting duplicates and ambiguities from the requirements, saving them in the product backlog.

3.2 Product Backlog

The product backlog is yet another artifact of Scrum. It serves as a collection of features for a product that are finished but require further components. The product owner assigns priority to the items in the product backlog; hence, the team begins by focusing on the most important feature. The sprint methodology and user stories, which provide concise explanations of functionality from the client's perspective for analyzing requirements, are the most common and effective strategies for

establishing a product backlog. Common and variable requirements are segregated in SPL, and the feature model is utilized for this purpose.

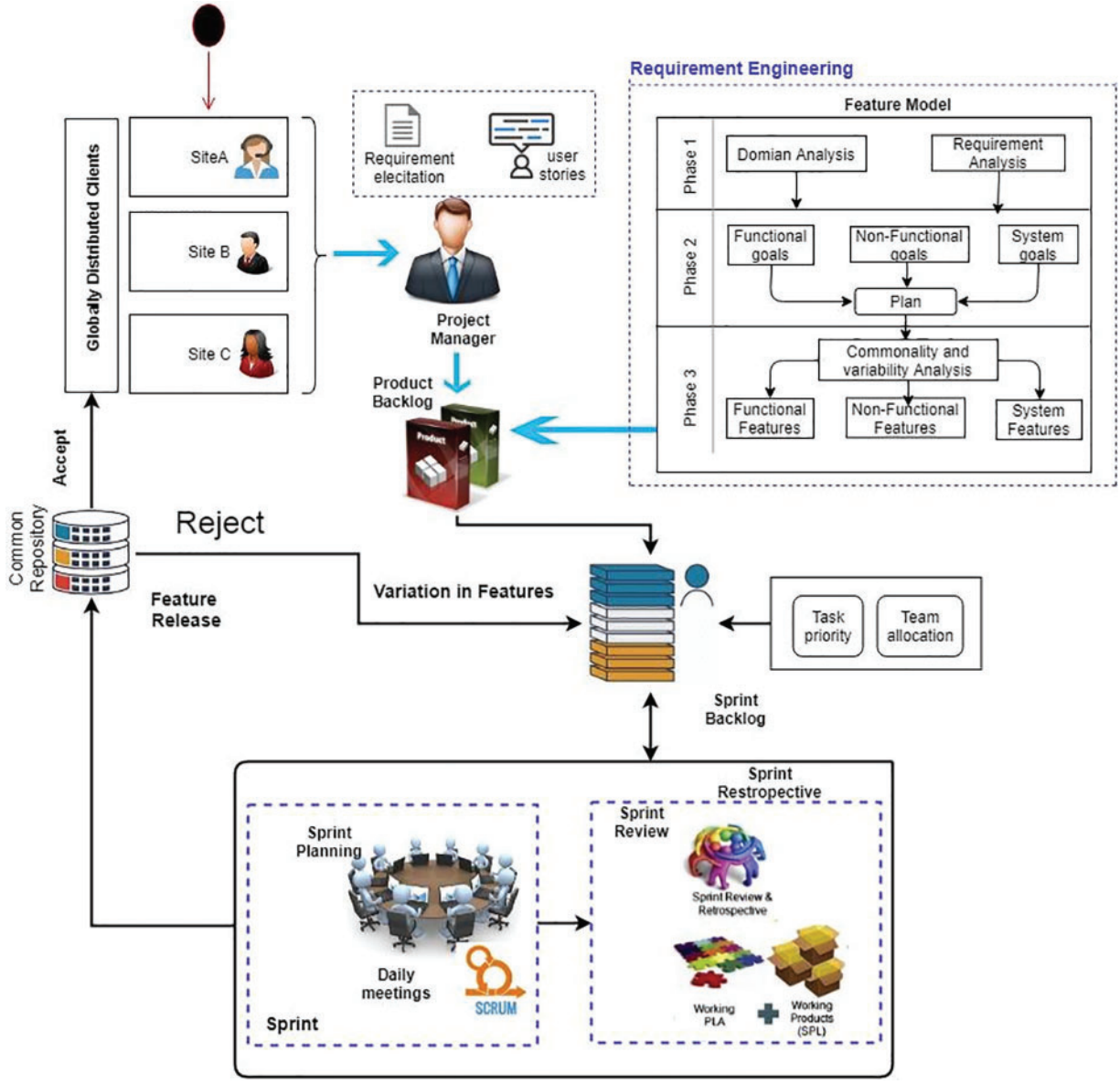

Figure 1: Proposed framework

3.3 Feature Model

The feature model conservatively represents all SPLs products in terms of "features." The feature diagram visually represents feature models. During the SPL development process, the feature model is widely used. To produce assets, such as reports, bits of code, and architecture definitions, preprocessing is commonly featured as input. Before the creation of a feature model, data is preprocessed in ML for structuring and extracting useful information. The main steps of preprocessing are depicted in Fig. 2.

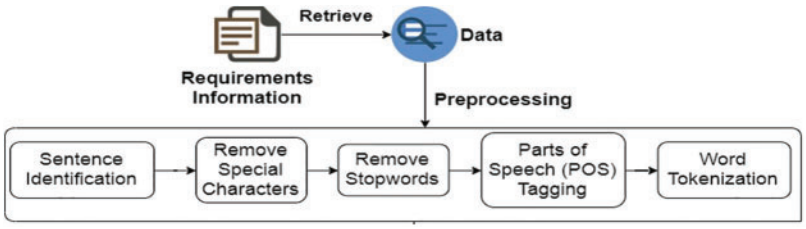

Figure 2: Preprocessing of information

3.3.1 Phase 1

In the feature model, requirements are analyzed, firstly, that are elicited from the globally distributed client and domain expert to analyze the domain and requirements. To separate functional, non-functional, and system aspects Weka tool is used. Weka is freely available software. The J48 classifier from Weka is built using the C4.5 decision tree development technique. Both filters and classifiers adhere to a hierarchy. The full name of the algorithm is "Weka classifiers trees J48" [3]. The results after the prioritization process show that the Weka tool's algorithm was utilized to classify needs.

These criteria were classified using the Weka tool's decision tree classifier based on the J48 algorithm. It was used to generate a tree of requirements that began at the root nodes and progressed through various leaves and sub-leaves until all required relationships or dependencies were discovered. It facilitates learning using a machine learning algorithm that is effective, error-free, and simple.

3.3.2 Phase 2

In phase 2, requirement analyst analyzes the commonality and variability among functional, non-functional, and system goals.

3.3.3 Phase 3

When analyzing common and variable features, functional, nonfunctional, and system features are separated based on common and variable requirements, and then saved in the product backlog. In Scrum project management, the sprint backlog is created by team members during planning meetings on the first day of the sprint. A product backlog is a list of features to be produced in the form of user stories, whereas a sprint backlog may be thought of as the team's to-do list for the sprint. In our project, the manager first determines the tasks that the team must do in order to offer the functionality it agreed to deliver during the sprint to Ireland, Australia, and Belgium based on client demands. These tasks are to be built in the first sprint.

The list of tasks that are in the sprint backlog is prioritized, and in this model, a common TFS repository is used. Source code management, reporting, requirements management, project management (for both agile and waterfall teams), automated builds, lab management, testing, and release management are all capabilities offered by the Microsoft product known as TFS. Additionally, TFS's common repository is used for feature release, and the Scrum cycle restarts if a customer requests any changes to be saved in the sprint backlog. After the identification of the feature model by classification of commonality and variability, components' features dependency is extracted using Algorithm 1. The algorithm's input is a list of features, and its output is a list of the least dependent components. Initial values for dependent variables are null at the beginning.

Algorithm 1: Selection of suitable components

Input: FM (List of Features)

Output: M_{LD} (List of Least Dependent Module)

1. FM: $\{F_1, F_2, F_3, \ldots, F_n\}$
2. Components: $\{C_1, C_2, C_3, \ldots, C_m\}$
3.
4. Components_Dep$\leftarrow$ N //Assign Dependency Value
5. $C_{Suit} \leftarrow \theta$
6. $C_{Sel} \leftarrow \theta$
7. $C_{CLD} \leftarrow \theta$
8. For each $(f \subseteq FM)$
9. **For each** $c \in Components$
10. **if** $(f \in c)$
11. **then** $C_{Suit} \leftarrow C_{Suit} \cup c$
12. **End For**
13. **End For**
14. For each $s \in C_{Suit}$
15. **if** $(s < Components\_Dep)$
16. **then** $C_{Sel} \leftarrow C_{Sel} \cup s$
17. **End For**
18. For each $y \in C_{Sel}$
19. **For each** $y \in C_{CLD}$
20. **if** $x < y$
21. **then** $C_{CLD} \leftarrow C_{CLD}x$
22. $C_{CLD} \leftarrow y$
23. **End For**
24. **End For**
25. Return C_{CLD}

The purpose is to discover the MLD among FM and components for the correct selection of components. It goes over each feature in the FM and compares it to each component in components to ensure a good fit. The components with the lowest dependency values are chosen and added to the list of least dependent modules.

Therefore, the PF for enhancing secure development in GSD for SPL provides a comprehensive method for combining state-of-the-art agile ideas with traditional development processes. By utilizing one platform, TFS, as a shared repository, PF organizes collaboration and communication between geographically dispersed teams and clients to mitigate the challenges of SPL. When TFS-integrated technologies are used, such as Microsoft Visual Studio and Eclipse, cross-platform cooperation runs easily, and separate development tools are no longer needed. Furthermore, in line with agile methodologies such as Scrum, PF incorporates a scientific approach to feature modeling, backlog management, and demand elicitation. By using ML methods for preprocessing and classification, PF ensures efficient handling of SPL through phases devoted to examining variability and similarity in demands. Moreover, agile project management techniques are made possible by the integration of a sprint backlog into TFS, which helps teams effectively prioritize work and adjust to changing customer expectations. Overall, by offering a methodical and consistent approach to software development

inside GSD contexts, PF manages the integration of agile concepts with conventional processes, optimizing output and user satisfaction.

4 Results and Discussion

We evaluated our proposed framework (PF) by performing an experiment and discussing the results using the following research hypotheses:

H0: PF is not better than the existing method. **H01:** PF is better than the existing method.

The PF is used to improve variability management based on requirements semantic analysis for feature model creation. It then classifies features into three categories: Functional, non-functional, and system features for the correct selection of feature components to identify variable and common features using machine learning. Traditional methods mostly focus on structured processes and do not correctly select features due to ambiguous requirements, communication, and coordination issues during SPL development in GSD within Agile. Whereas ML is used for adaptability and probabilistic reasoning. Therefore, ML integration during development considers data preprocessing, classification, and deployment for DVM.4.1

4.1 Industrial Case Study

Due to company confidentiality, we did not mention the name and details of all the projects used for the experiment and comparison. For evaluation, we performed case study on multiple projects and chose one company working in the SPL domain in a GSD environment, XYZ Technologies Company (the name of the company is not mentioned due to privacy restrictions). The company's head office is in Pakistan, and it also has branches in South Africa, New Zealand, Qatar, the United Arab Emirates, and the United Kingdom. The company has around 16 employees at its head office in Pakistan, and approximately 300 employees worldwide.

We worked on four different projects, with two focusing on Android applications and two in another domain. Due to confidentiality, we summarize the results in two projects to evaluate the PF performance and practicability in a real scenario. The company has different projects to its credit, which include version-based and SPL-based projects. There are two groups: Group 1 (G1) worked on Project-A (P-A), and Group 2 (G2) worked on Project-B (P-B).

Out of the large set of projects, P-A involves developing a product family for an Android app for the digital Quran, aiming to increase customer satisfaction by offering easy-to-use features at a moderate cost. The first product includes the ability to read Quran pages by pressing forward and backward buttons. Another product in this line has an optional sound function. A third product includes a Quran translation feature, while a fourth product adds a Tafseer of the Quran feature.

P-B is an Android photo camera app development project of a product family, enabling various photo filters during photo capture for clients, providing different filters for a beautiful image that a real camera does not provide. Stickers on images are an extra feature of P-B's second product. Furthermore, another product from this family offers the additional feature of adding music to videos.

During the development of the GSD product family, XYZ Technologies Company used a variety of tools and applications to fill in the gaps in variability management, communication, control, and coordination. They agreed to utilize our PF for development because its main objectives include achieving reusability of artifacts and providing quick and effective product development.

4.2 Control Experiment (CE)

For the purpose of validating performance using the With Existing Approach (WEA), a CE was held prior to executing the PF. A set of 16 participants from P-A and P-B were involved in the PF execution to control SPL commonality and variability, and to facilitate communication and coordination among ABC Technologies Company's geographically scattered employees. In Fig. 3, additional categories for these 16 members are shown.

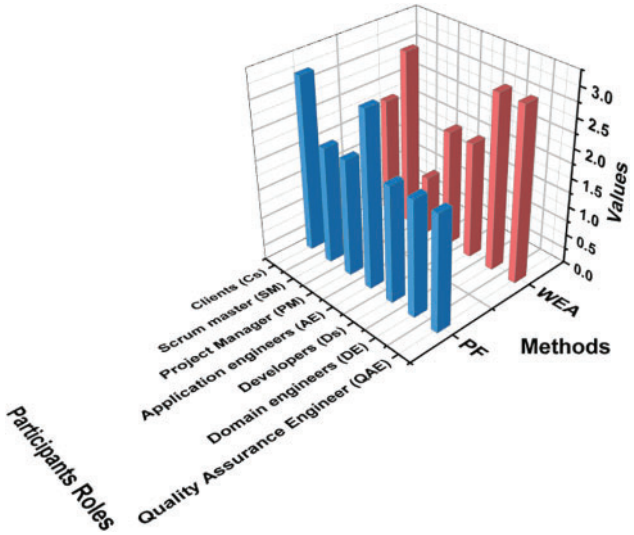

Figure 3: Participants for evaluation

The complete execution of the PF and WEA involved utilizing both strategies in a similar domain. For project execution, a set of key assets and variabilities were chosen at the start of the project and during the changing of product features. Both G1 and G2 used our PF, which utilizes TFS as a single repository to address communication, coordination, and control issues, as well as cutting-edge techniques like Scrum to improve reuse in GSD. Using TFS services on a separate platform, the entire team membership was connected. In SPL, there is no mandate to use particular networking techniques.

Every person was involved in the execution of this PF, and they all provided extremely clear and straightforward TFS services to assist with the crucial tasks of communication coordination and variability management in worldwide distributed software development. Decisions were made after viewing the change management report by all members.

4.3 WEA

All team members of G2 were included during the execution of WEA. The presently used conventional methodology made use of numerous locations with diverse logins, and people did not have access to software development life cycles. Due to the fact that certain venues do not require overlapping workdays or lengthy meetings, they are not feasible. The usage of the English language is also restricted by this traditional approach. This may result in expense, uncertainty, incompleteness, effort, time, and other concerns such as cultural differences and language. The conventional current approach used a waterfall approach for product family creation.

After executing both methodologies, we concluded that our PF is understandable, executable, and improves correspondence, control, and coordination between the software team and stakeholders during development. Our research hypothesis states that in order to build product lines in GSD, PF will be able to handle issues with collaboration, communication, stakeholders, or clients' control as well as the GSD developing team along with software development teams. The criteria for evaluation depend upon challenges and issues such as dynamic variability management, secure development, and correct selection of features in SPL during agile-based GSD, which were identified from existing studies [3,5,19,34,35], including (depicted in Fig. 4) communication, reusability, repository management, coordination, control, usability, and completeness as mentioned in Table 1. These matrices evaluate the performance of PF in comparison to WEA by adopting semantic analysis, repository management, and one platform by integrating different platforms and using classification and selection algorithms for DVM in GSD. We can conclude that our PF outperforms WEA. As a result, we conducted interviews with participants from both groups to examine various elements represented and explained in Fig. 4. As a result, it maximizes the utilization of available resources while also producing high productivity.

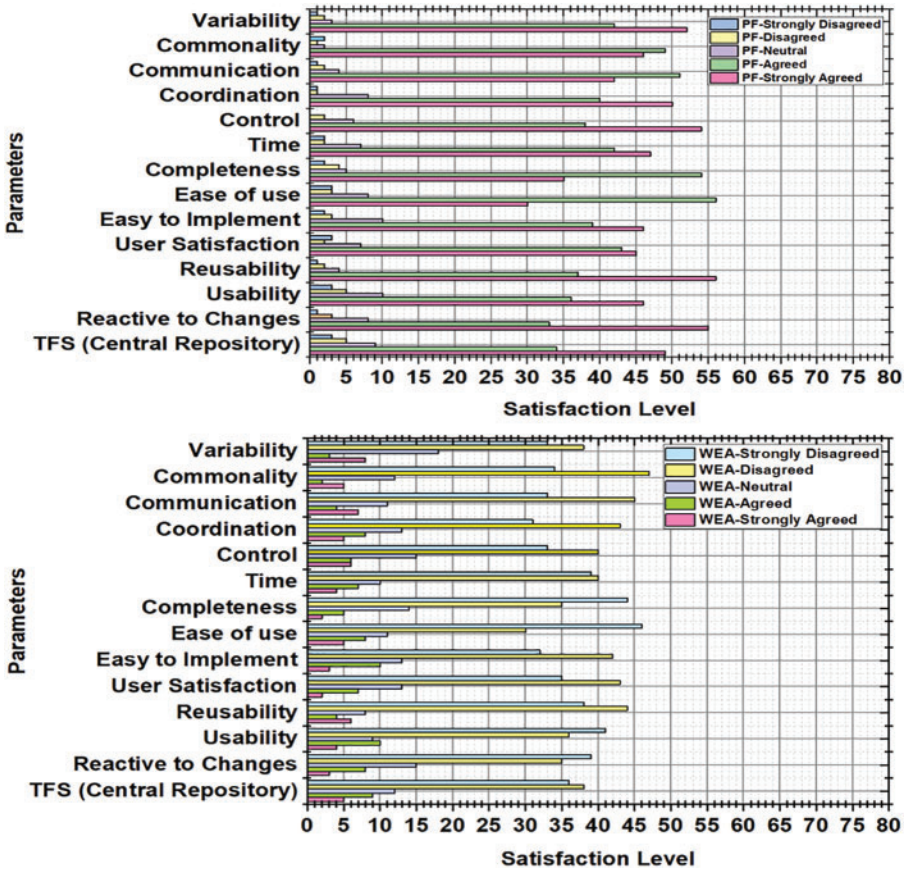

Figure 4: Performance analysis

While the y-axis indicates features, the x-axis represents components. The colors of the cells in Fig. 4 indicate the level of satisfaction with TFS Central Repository's (TFSCR) capacity to modify variability for each particular feature and component. Blue cells represent user satisfaction

with TFSCR's ability to modify the variability of that feature and component, while red cells indicate customer dissatisfaction. The least dependent modules, such as TFS (Central Repository), Completeness, Ease of Use, Usability, Reusability, and User Satisfaction, are shown in the diagram. These components are the most satisfied with the TFSCR they received to adjust variability. The graph also illustrates which attributes—time, control, coordination, communication, commonality, and variability—are most reliant on TFSCR. TFSCR receives the least satisfaction for variability change, suggesting that this aspect can be improved.

TFSCR, the least dependent module, has a 90 percent satisfaction rating, indicating that users are satisfied with TFSCR's handling of component updates. Both "Usability" and "Reusability" receive 80% satisfaction, suggesting that TFSCR is easy to use and that users can reuse components. However, TFSCR only achieves 20% satisfaction for "Time" and "Control," indicating user dissatisfaction with its handling of feature updates. The graph also reveals that as feature complexity increases, TFSCR satisfaction declines, suggesting that TFSCR performs better with simpler features than with complicated ones. This information helps in determining and prioritizing areas for improvement. Fig. 5 describes the participants in Groups 1 and 2's levels of satisfaction. Group 1 participants, who utilized PF, are shown to be happier than Group 2 participants who used WEA. The PF demonstrates more efficiency and cost-effectiveness compared to WEA. We employed SPSS 17.0 statistical analysis to discover significant differences between WEA and PF. Reliability analysis was conducted to ensure the accuracy of our findings. Cronbach's Alpha, the most popular reliability analysis metric, was used to assess the accuracy of the measurement scale, yielding a value of 0.965, indicating high reliability. Factor analysis was also conducted to uncover hidden variables, further ensuring the trustworthiness of our findings. Table 2 provides an overview of case processing.

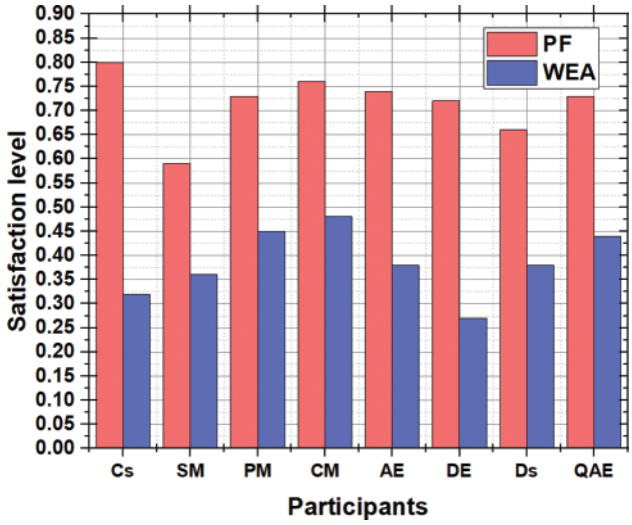

Figure 5: Approach comparison

The KMO test determines whether the sample size is sufficient by displaying the percentage of variance in the variables. To assess whether samples from a population have equal variance, Bartlett's test is applied. The test results from KMO and Bartlett are displayed in Table 3, which demonstrates the variance and sufficiency with values of 0.650 and 0.021, respectively. The combined insights result as appeared in Fig. 6, showing that the standard deviation (SD) and standard error mean (SEM) values

of PF (i.e., 8.51, 3.20) are less variable and more reliable than WEA (i.e., 10.33, 3.53). This indicates that the values of PF are less scattered from their mean (M) values.

Table 2: Case processing summary

		N	%
Cases	Valid	2	100
	Excluded[a]	0	0.0
	Total	2	100

Note: a. All variables in the method are deleted list wise.

Table 3: KMO and Bartlett's test

Kaiser-Meyer-Olkin measure of sampling adequacy		0.650
Bartlett's sphericity test	Approximately chi-square	2.16
	Difference	1
	Significance	0.021

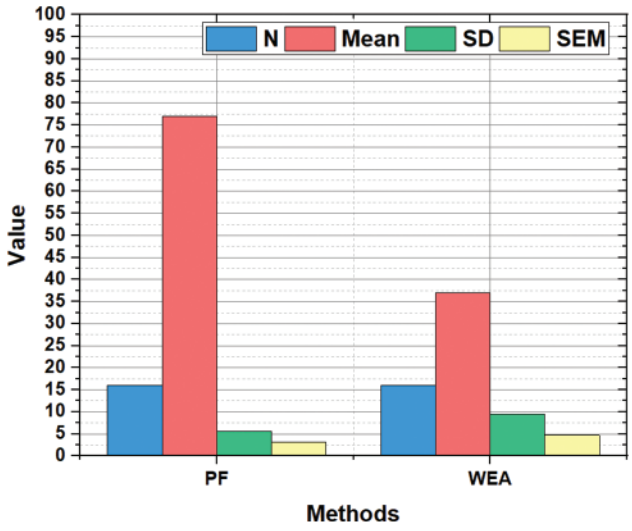

Figure 6: Paired sample statistics

Table 4 demonstrates that the mean distinction between groups is not equivalent to zero, and the p-value is less than the significance level (i.e., $\alpha = 0.05$ or 5%). Thus, the alternative hypothesis is accepted, and the null hypothesis is rejected.

The results of a paired samples test that compares the performance differences between PF and WEA are shown in Table 4. The mean difference between PF and WEA is 39.56, as seen in the "Pair Difference" column. The t-value, which is 7.3 and measures the difference standardized by the standard error, is shown in the "t" column. The degrees of freedom, which in this case are eight, are displayed in the "df" column. The p-value, a measure of the evidence against a null hypothesis, is shown in

the "Significance (2-tailed)" column. The *p*-value in this case is 0.003, which is lower than the usual 0.05 threshold of significance. The low *p*-value indicates the statistical significance of the observed differences between PF and WEA.

Table 4: Paired samples test

		Pair difference					t	df	Significance (2-tailed)
		M	SD	SEM	95% CID				
					Lower	Upper			
Pair	PF-WEA	39.56	19.8	4.98	29.76	61.2	7.3	8	0.003

The SD of the differences, 19.8, is represented by the "SD" column, while the mean of the differences, 39.56, is represented by the "M" column. The standard error of the mean, 4.98, is shown in the "SEM" column. The columns labeled "95% CID" (Confidence Interval for the Difference) offer a range indicating the likelihood that the actual difference between PF and WEA lies inside. This confidence interval has lower and upper bounds of 29.76 and 61.23, respectively. The exclusion of zero from the interval adds credence to the notion that the two methods differ significantly from one another.

As a whole, paired samples test findings indicate that PF performs significantly better than WEA, with a confidence interval that does not span zero. This research provides a thorough evaluation of complex problems confronting SPLE in the context of GSD. The study sets itself apart by delving deeply into specific shortcomings in state-of-the-art methods, painstakingly addressing problems such as large-scale conversions, feature modeling, scoping, and quality assurance. It also investigates anticipated SPLE developments, offering a progressive viewpoint by highlighting vital requirements for protecting sensitive data in the GSD process.

This academic discourse on PF contributes to increasing practitioners' real-world knowledge for improving the efficiency and security of SPLE activities in dynamic GSD during requirements specification, feature identification, selection, and prioritization processes. The study makes a significant contribution to academic and professional sectors for SPLE in GSD. The combined requirement and testing methods directly address the challenges of managing unpredictability in GSD. By incorporating testing early in the development process and integrating it into requirement validation, our PF enables proactive detection and resolution of variability-related issues. This proactive technique decreases the possibility of miscommunication and misunderstanding across distributed teams, thus improving the overall efficiency and efficacy of DVM. Furthermore, by emphasizing the practical implications of our PF, such as enhanced collaboration and shorter development timeframes, it addresses real-world challenges in SPL and GSE.

For the evaluation of PF to mitigate identified challenges, we performed an experiment on real-world projects for its practical implication in XYZ organization to ensure reliability and resilience for DVM and secure development using ML during GSD based SPL. Participants with experience in SPL development in GSD, such as quality engineers, developers, and domain engineers, were included. They participated in PF and WEA implementation, and the results showed that the satisfaction level of PF participants significantly increased due to the significant improvement of PF performance for secure development and DVM using ML compared to WEA participants' satisfaction level. Feedback

from the participants was extracted using questionnaires based on matrices selected for evaluation as mentioned in the experiment setup.

In this research, firstly we managed DV features using semantic analysis of requirements then categorizing features into functional, non-functional, and system features using ML J48 model selected correct components. After that, priority was assigned to features for correct allocation of tasks, development, and software validation. Thus, for the evaluation of PF, we conducted an industry case study at GSD's XYZ Technologies Company, which specializes in SPL. We assessed PF's performance using both qualitative and quantitative indicators, using experiments. The findings of this study shows that PF deployment considerably increases collaboration, communication, and variability control in GSD. Furthermore, we addressed potential challenges to validity, ensuring that our experimental findings were reliable and resilient. Overall, research shows that combining Agile methodology concepts with tools can improve the development of SPLs in a GSD.

5 Validity of Threats

This section discusses potential threats to the validity of experiments conducted in accordance with the recommendations provided in previous studies [2,3,13].

5.1 Construct Validity

The correct operational measurement of required infrastructure is a crucial problem in this threat. To resolve this issue, construct validity was established by using the same analysis methods for both process models. To further prevent bias, individuals were informed that their participation would not affect their grades. The experimental hypothesis was not disclosed to protect confidentiality and prevent researcher bias.

5.2 Internal Validity

This threat concerns the possibility that experts may be biased when interacting with participants in the experiment. To mitigate this threat, the experiment was carefully conducted, and volunteers were provided with detailed tutorials and labs to thoroughly test the proposed framework. Random groups were formed to reduce bias, and participants were assured that their work would not be affected by bias. Participants were encouraged to share their thoughts and ideas to ensure their full participation in the experiment.

5.3 External Validity

The main concern with this threat is the generalizability of the experiment's results to other situations. To address this issue, participants from real businesses were included in the experiment. However, the results of the experiment may not hold true when considering security risks in SPL development. The introduction of security threats could alter the replication findings.

5.4 Conclusion Validity

If the assumptions of the statistical test are violated, a statistically significant result may not truly indicate a relationship between the variables. Because the trial data is on an interval scale, it may not be appropriate to use statistical testing to improve results. To address this issue, the non-parametric Mann-Whitney U test, which does not rely on assumptions about the distribution of data, was utilized.

While the sample size is sufficient for the statistical test, it is not excessively large. A larger sample size would increase the test's power.

6 Conclusion

This research identifies and addresses the primary challenges encountered in developing a SPL within a global context. Managing core assets, commonality, and variability emerged as significant hurdles. Additionally, difficulties arose from the absence of a centralized database, coordination issues, and communication gaps, particularly in the realm of change management. To address these challenges, we propose a comprehensive framework that seamlessly integrates Agile methodologies with TFS. The aim of this integration is to effectively manage variability and commonality in product line development within a global environment, while simultaneously minimizing time and expenses. Through rigorous experimentation and comparison with traditional methods, our framework has demonstrated significant improvements in performance, communication, coordination, and risk mitigation. These improvements are observed throughout the development lifecycle. Looking ahead, our future plans involve leveraging component-based software in GSD. We also prioritize testing of SPL post-requirement changes to enhance SPL validation in GSD during DVM. Furthermore, to enhance the efficacy of product families within DVM, we are committed to actively seeking feedback from practitioners. We also aim to integrate cutting-edge technologies such as artificial intelligence and deep learning [36]. In conclusion, our research delves into the challenges associated with establishing SPL within a flexible GSD framework and offers invaluable insights for researchers, developers, and testers striving to enhance secure development practices in GSD projects.

Acknowledgement: We would like to thank the anonymous referees for making several recommendations that have improved our paper.

Funding Statement: This study is supported via funding from Ministry of Defense, Government of Pakistan under Project Number AHQ/95013/6/4/8/NASTP(ACP). Titled: Development of ICT and Artificial Intelligence Based Precision Agriculture Systems Utilizing Dual-Use Aerospace Technologies-GREENAI.

Author Contributions: Conceptualization, Marya Iqbal, Yaser Hafeez and Mamoona Humayun; Formal analysis, Nabil Almashfi, Amjad Alsirhani, Faeiz Alserhani; Funding acquisition, Nabil Almashfi; Project administration, Sadia Ali; Supervision, Yaser Hafeez and Mamoona Humayun; Validation, Nabil Almashfi, Amjad Alsirhani, Faeiz Alserhani; Writing–original draft, Marya Iqbal; Writing–review & editing, Marya Iqbal, Sadia Ali and Muhammad Jamal.

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

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

References

[1] M. Bhushan, A. Negi, P. Samant, S. Goel, and A. Kumar, "A classification and systematic review of product line feature model defects," *Softw. Qual. J.*, vol. 28, no. 4, pp. 1507–1550, Dec. 2020. doi: 10.1007/s11219-020-09522-1.

[2] A. Ali *et al.*, "A data mining technique to improve configuration prioritization framework for component-based systems: An empirical study," *Inf. Technol. Control.*, vol. 50, no. 3, pp. 424–442, Sep. 2021. doi: 10.5755/j01.itc.50.3.27622.

[3] A. A. Kiani, Y. Hafeez, M. Imran, and S. Ali, "A dynamic variability management approach working with agile product line engineering practices for reusing features," *J. Supercomput.*, vol. 77, no. 8, pp. 8391–8432, Aug. 2021. doi: 10.1007/s11227-021-03627-5.

[4] G. Giray, "A software engineering perspective on engineering machine learning systems: State of the art and challenges," *J. Syst. Softw.*, vol. 180, no. 4, pp. 111031, Oct. 2021. doi: 10.1016/j.jss.2021.111031.

[5] C. Gupta and V. Gupta, "A decentralized framework for managing task allocation in distributed software engineering," *Appl. Sci.*, vol. 11, no. 22, pp. 10633, Nov. 2021. doi: 10.3390/app112210633.

[6] A. Olyai and R. Rezaei, "Analysis and comparison of software product line frameworks," *J. Softw.*, vol. 10, no. 8, pp. 991–1001, 2015. doi: 10.17706/jsw.10.8.991-1001.

[7] G. Bakirtzis, B. T. Carter, C. R. Elks, and C. H. Fleming, "A model-based approach to security analysis for cyber-physical systems," in *2018 Annu. IEEE Int. Syst. Conf. (SysCon)*, Vancouver, BC, Canada, IEEE, Apr. 2018, pp. 1–8. doi: 10.1109/SYSCON.2018.8369518.

[8] S. Zhao, Q. Zhang, Z. Peng, and Y. Fan, "Integrating customer requirements into customized product configuration design based on Kano's model," *J. Intell. Manuf.*, vol. 31, no. 3, pp. 597–613, Mar. 2020. doi: 10.1007/s10845-019-01467-y.

[9] K. Imran, N. Anjum, A. Alghamdi, A. Shaikh, M. Hamdi and S. Mahfooz, "A secure and efficient cluster-based authentication scheme for internet of things (IoTs)," *Comput. Mater. Contin.*, vol. 70, no. 1, pp. 1033–1052, 2022. doi: 10.32604/cmc.2022.018589.

[10] K. Hayashi, M. Aoyama, and K. Kobata, "Agile tames product line variability: An agile development method for multiple product lines of automotive software systems," in *Proc. 21st Int. Syst. Softw. Prod. Line Conf.*, Sevilla, Spain, ACM, Sep. 2017, pp. 180–189. doi: 10.1145/3106195.3106221.

[11] L. Tan and Y. Lin, "An aspect-oriented feature modelling framework for software product line engineering," in *Proc. ASWEC, 2015 24th Australas. Softw. Eng. Conf.*, Adelaide, SA, Australia, ACM, Sep. 2015, pp. 111–115. doi: 10.1145/2811681.2811703.

[12] J. M. Bass, "Artefacts and agile method tailoring in large-scale offshore software development programmes," *Inf. Softw. Technol.*, vol. 75, no. 9–10, pp. 1–16, Jul. 2016. doi: 10.1016/j.infsof.2016.03.001.

[13] S. Ali, Y. Hafeez, S. Asghar, A. Nawaz, and S. Saeed, "Aspect-based requirements mining technique to improve prioritisation process: Multi-stakeholder perspective," *IET Softw.*, vol. 14, no. 5, pp. 482–492, Oct. 2020. doi: 10.1049/iet-sen.2019.0332.

[14] J. Klünder, P. Hohl, and K. Schneider, "Becoming agile while preserving software product lines: An agile transformation model for large companies," in *Proc. 2018 Int. Conf. Softw. Syst. Process*, Gothenburg, Sweden, ACM, May 2018, pp. 1–10. doi: 10.1145/3202710.3203146.

[15] J. Martinez, T. Ziadi, T. F. Bissyandé, J. Klein, and Y. Le Traon, "Bottom-up adoption of software product lines: A generic and extensible approach," in *Proc. 19th Int. Conf. Softw. Prod. Line*, Nashville, Tennessee, ACM, Jul. 2015, pp. 101–110. doi: 10.1145/2791060.2791086.

[16] A. Al-Hawari, H. Najadat, and R. Shatnawi, "Classification of application reviews into software maintenance tasks using data mining techniques," *Softw. Qual. J.*, vol. 29, no. 3, pp. 667–703, Sep. 2021. doi: 10.1007/s11219-020-09529-8.

[17] S. Ali, Y. Hafeez, M. Humayun, N. Z. Jhanjhi, and D. N. Le, "Towards aspect based requirements mining for trace retrieval of component-based software management process in globally distributed environment," *Inf. Technol. Manag.*, vol. 23, no. 3, pp. 151–165, Sep. 2022. doi: 10.1007/s10799-021-00343-7.

[18] V. Bauer and A. Vetro', "Comparing reuse practices in two large software-producing companies," *J. Syst. Softw.*, vol. 117, no. 4, pp. 545–582, Jul. 2016. doi: 10.1016/j.jss.2016.03.067.

[19] M. Daniel, N. Lawrence, and K. Michael, "Embedding quality into software product line variability artifacts," *Int. J. Softw. Eng. Appl.*, vol. 12, no. 3, pp. 11–25, May 2021. doi: 10.5121/ijsea.2021.12302.

[20] J. M. Horcas, M. Pinto, and L. Fuentes, "Empirical analysis of the tool support for software product lines," *Softw. Syst. Model.*, vol. 22, no. 1, pp. 377–414, Feb. 2023. doi: 10.1007/s10270-022-01011-2.

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

[22] I. Reinhartz-Berger and S. Abbas, "Extracting domain behaviors through multi-criteria, polymorphism-inspired variability analysis," *Inf. Syst.*, vol. 108, no. 6, pp. 101882, Sep. 2022. doi: 10.1016/j.is.2021.101882.

[23] G. Adamson, L. Wang, and P. Moore, "Feature-based control and information framework for adaptive and distributed manufacturing in cyber physical systems," *J. Manuf. Syst.*, vol. 43, no. 2, pp. 305–315, Apr. 2017. doi: 10.1016/j.jmsy.2016.12.003.

[24] J. Javaid, A. A. Kiani, Y. Hafeez, N. Iltaf, F. B. Ahmed and G. Abbas, "Improving software product line challenges in globally distributed environment using modern strategies," in *2023 Int. Conf. Commun. Technol. (ComTech)*, Rawalpindi, Pakistan, IEEE, Mar. 2023, pp. 134–139. doi: 10.1109/ComTech57708.2023.10165120.

[25] A. B. Harris, "Exploring the agile system development best practices cybersecurity leaders need to establish a cyber-resilient system: A phenomenological study," Ph.D. dissertation, Colorado Technical University, USA, 2019.

[26] O. Aguayo, S. Sepúlveda, and R. Mazo, "Variability management in self-adaptive systems through deep learning: A dynamic software product line approach," *Electronics*, vol. 13, no. 5, pp. 905, Feb. 2024. doi: 10.3390/electronics13050905.

[27] M. Babar, M. H. Roos, and P. H. Nguyen, "Machine learning for agile and self-adaptive congestion management in active distribution networks," in *2019 IEEE Int. Conf. Environ. Electr. Eng. 2019 IEEE Indust. Comm. Power Syst. Europe (EEEIC/I&CPS Europe)*, IEEE, Jun. 11, 2019, pp. 1–6. doi: 10.1109/EEEIC.2019.8783624.

[28] S. Hooda, V. M. Sood, Y. Singh, S. Dalal, and M. Sood, *Agile Software Development: Trends, Challenges and Applications*. Hoboken, NJ, USA: John Wiley & Sons, 2023.

[29] R. Hanslo and M. Tanner, "Machine learning models to predict agile methodology adoption," in *2020 15th Conf. Comp. Sci. Infor. Sys. (FedCSIS)*, IEEE, Sep. 6, 2020, pp. 697–704.

[30] N. Zainuddin, A. Selamat, and R. Ibrahim, "Improving twitter aspect-based sentiment analysis using hybrid approach," in N. T. Nguyen, B. Trawiński, H. Fujita, and T. P. Hong (Ed.), *Intell. Inf. Database Syst.*, Berlin, Heidelberg: Springer, 2016, vol. 9621, pp. 151–160. doi: 10.1007/978-3-662-49381-6_15.

[31] A. Metzger and K. Pohl, "Software product line engineering and variability management: Achievements and challenges," in *Future Softw. Eng. Proc.*, Hyderabad India, ACM, May 2014, pp. 70–84. doi: 10.1145/2593882.2593888.

[32] K. Sagar and A. Saha, "A systematic review of software usability studies," *Int. J. Inf. Tecnol.*, Dec. 2017. doi: 10.1007/s41870-017-0048-1.

[33] S. Mahmood, S. Anwer, M. Niazi, M. Alshayeb, and I. Richardson, "Key factors that influence task allocation in global software development," *Inf. Softw. Technol.*, vol. 91, pp. 102–122, Nov. 2017. doi: 10.1016/j.infsof.2017.06.009.

[34] A. Yasin, R. Fatima, J. Ali Khan, L. Liu, R. Ali and J. Wang, "Counteracting sociocultural barriers in global software engineering using group activities," *J. Softw. Evol. Process*, vol. 36, no. 5, pp. e2587, 2024. doi: 10.1002/smr.2587.

[35] A. Yasin, R. Fatima, J. B. Zheng, W. Afzal, and S. Raza, "Can serious gaming tactics bolster spear-phishing and phishing resilience?: Securing the human hacking in information security," *Inf. Softw. Technol.*, vol. 170, no. 2, pp. 107426, 2024. doi: 10.1016/j.infsof.2024.107426.

[36] Y. Yang, X. Xia, D. Lo, and J. Grundy, "A survey on deep learning for software engineering," *ACM Comput. Surv.*, vol. 54, no. 10s, pp. 1–73, 2022. doi: 10.1145/3505243.