Pertanika J. Sci. & Technol. 34 (4): 2255 - 2273 (2026)

PERTANIKA JOURNALS

SCIENCE & TECHNOLOGY

Journal homepage: http://www.pertanika.upm.edu.my/

A Framework for Secure Software Engineering in Hybrid Agile Software Development

Normi Sham Awang Abu Bakar [1]*, Norzariyah Yahya[1], and Siti Sarah Maidin[2]

[1]Department of Computer Science, Faculty of ICT, International Islamic University Malaysia, 53100 Gombak, Kuala Lumpur, Malaysia

[2]Faculty of Data Science and Information Technology, INTI International University, 71800 Nilai, Negeri Sembilan, Malaysia

ABSTRACT

This study examines how Secure Software Engineering (SSE) practices can be applied while developing with Hybrid Agile methods. The Hybrid Agile approach combines disciplined planning and extensive documentation, which are the hallmarks of traditional software development, with flexibility, incremental delivery and quick adoption for changing requirements contributed by Agile practices. Generally, existing secure software engineering frameworks only support either classical or Agile methods. As a result, Hybrid Agile implementations rarely provide enough guidance to integrate security throughout the full development life cycle. The purpose of this research is to investigate the current utilisation of secure software engineering practices in Hybrid Agile to identify and introduce security-focused coding elements designed for those environments. In line with the qualitative research approach being used, we evaluate current secure software practices in Hybrid Agile project implementation. In this study, interviews with professional programmers in diverse business sectors were conducted. It is evident that there is a lack of standardised security practices and that it is crucial for managers and project managers to encourage and support these. In addition, it is important that this study highlights the importance of automation tools in secure software engineering practices. With these findings in mind, this paper provides a structured approach for these organisations seeking to integrate security into Hybrid Agile projects and for improving the reliability and integrity of their software applications. The shift towards proactive environments in hybrid software applications requires an assessment of their software development life cycle to deal with security weaknesses and build tools for detecting possible threats in their applications.

Keywords: Hybrid agile, qualitative analysis, secure software engineering, Software Development Lifecycle (SDLC), software security framework

ARTICLE INFO

Article history:
Received: 17 November 2025
Accepted: 12 June 2026
Published: 20 August 2026

DOI: https://doi.org/10.47836/pjst.34.4.14

E-mail addresses:
nsham@iium.edu.my (Normi Sham Awang Abu Bakar)
norzariyah@iium.edu.my (Norzariyah Yahya)
sitisarah.maidin@newinti.edu.my (Siti Sarah Maidin)
* Corresponding author

e-ISSN: 2231-8526

INTRODUCTION

In the contemporary business environment, there are always attempts by business structures to conceptualise and implement strategies or techniques of producing the highest output and quality, as well as improving the response to changes within the marketplace. The Hybrid Agile approach hinges on traditional project management practices and considers the flexible spirit of Agile methodologies. Such an approach ensures well-rounded software development by applying the ability to shift emphasis and improve, which lies within the Agile approach.

In conventional methods, the security measures are very strict, but their implementation is performed stage by stage, causing latency in the detection of the problem and, ultimately, lower agility towards the threats as well. Moreover, the emphasis on the implementation of various security updates is performed throughout the Agile iteration process, which enables earlier detection and rapid mitigation of security threats. The implementation requires specialised expertise and continuous security monitoring. The new approaches combine secure protocols within an Agile process, such as secure coding and continuous integration of security, but the existing Agile protection mechanisms are unable to fulfil the demands of Hybrid Agile as well.

Hybrid Agile methodologies, which combine the careful documentation strategy with flexibility, are useful for risk management and adaptability, but for execution purposes, need proper synchronisation and communication for managing the complexity (ISO/IEC, 2008; Prenner et al., 2021; Theocharis et al., 2015). In previous studies on secure software development methodologies, there have been developments made specifically for the context of Agile methodologies (Kagombe et al., 2021; Rindell et al., 2018). These secure-aggregated developments safeguard the combined execution of Agile software development methodologies, such as secure development methodologies, vulnerability management, and long-lasting protection strategies, but do not adequately focus on the unique challenges posed by the combination of Hybrid Agile methodologies for development with the options of conventional development methodologies. This leads to adaptability, but with structured and formalised process activities like Requirement Elicitation and System Design, which are comparatively more formalised than those in Agile methodologies.

Therefore, security systems aligned only with Agile may be inefficient for the broader need of Hybrid Agile, considering its inclusion of traditional procedural security requirements. Specifically, three critical research gaps can be identified in the extant literature. First, existing SSE frameworks such as SSE-CMM (ISO/IEC, 2008) and SAMM (Rindell et al., 2018) were conceived for either Waterfall-style plan-driven environments or pure Agile iterations; neither addresses the coexistence of formal phase gates (e.g., requirements elicitation, high-level architecture) with iterative sprint cycles that characterise Hybrid Agile. Second, the empirical understanding of how practising software developers

perceive, adopt, or resist security measures within Hybrid Agile contexts is severely limited, with only a handful of studies (Ding et al., 2020; Maidin & Yahya, 2024) exploring this domain, and none grounding their findings in primary data gathered through direct interviews with industry professionals. Third, the role of organisational and managerial factors such as policy enforcement, security training, and cross-functional collaboration in determining the success or failure of SSE integration in Hybrid Agile projects remains largely unexplored, despite the recognised importance of such socio-technical dimensions in technology adoption (Wang et al., 2024). Filling this research gap is paramount to ensure seamless integration of safe practices within Hybrid Agile strategies for optimised use of dependable and secure software outcomes. This feeds into the importance of research in safe coding practices.

Against this backdrop, the major contributions of this study are highlighted as follows:

- This study presents the first empirical investigation of Secure Software Engineering (SSE) practices specifically within the Hybrid Agile development context, addressing a gap that prior frameworks targeting either traditional or pure Agile environments have left unfilled.
- Through semi-structured interviews with ten experienced software developers across diverse Malaysian industries, this study identifies and documents the real-world challenges, tool preferences, and management practices that shape SSE adoption in Hybrid Agile projects.
- A Secure Hybrid Agile Framework is proposed that integrates security practices across all phases of the Hybrid Agile lifecycle from planning and estimation through iterative sprint development to final integration and system testing, offering practitioners actionable guidance for building secure software within this blended methodology.
- The study highlights key enablers of secure development in Hybrid Agile settings, including management enforcement of security policies, developer participation in the design of security procedures, deployment of automated security tools, and targeted security training, findings that carry direct implications for software engineering managers and project leads.

RELATED WORK

This section explores the related work as guidelines for the framework development.

Traditional Models

A traditional software development process model, known as the Waterfall Model, follows a linear and sequential methodology in which each stage of the process must be completed before proceeding to the next (Prenner et al., 2020). Projects with clearly defined

requirements and those where changes are unlikely to occur during development are best suited for the Waterfall Model. It is also suitable for smaller projects or those for which a precise deadline is necessary (Tell et al., 2019).

The Verification and Validation Model, or V-Model, is an expansion of the conventional Waterfall Model. The significance of validation and verification tasks at every level of the development lifecycle is emphasised, and a V-shaped process is created, with a testing phase for every development phase (Prenner et al., 2020). The V-Model is very helpful for projects where quality and dependability are very important, and the requirements are clear and consistent. The V-Model is suitable for projects in the aerospace, medical, and automotive sectors, among others, that have stringent regulatory and compliance standards (Tell et al., 2019).

Agile Models

Agile development methodology is characterised by its flexibility and inventiveness. It is a tactic for thriving in the chaos and uncertainty that software development projects frequently bring. Agile processes typically use evolutionary and iterative approaches to software development, which reduce the development cycle and improve the final product's quality (Asthana et al., 2012). Agile, as a lightweight process paradigm, emphasises cooperation and communication amongst all stakeholders. The main goals of agile development are to reduce overall costs, shorten development cycles, and improve software quality (Maier et al., 2017). Agile promotes evolutionary and iterative methods of development, which reduce the time needed to produce and deliver a product. In this work, we primarily concentrate on the Scrum model, one of the Agile models.

One of the popular agile methods for developing software is Scrum (Schwaber & Sutherland, 2016). A product owner, a Scrum master, and a development team make up a Scrum team. The project's goals, specifications, and release schedule are specified by the product owner. Every specified functional and non-functional requirement is kept in a product backlog. The Scrum master facilitates communication between the product owner and the development team and assists the team in adhering to Scrum practices. The goal of a development team is to implement the software; this group should ideally consist of three to nine members.

Hybrid Agile Models

The Waterfall and V-Model are examples of traditional software development models that provide a robust framework for long-term planning. However, these models lack the capability for rapid feedback, which can make it harder to avoid mistakes because the product is usually only delivered to the customer at the end of the project, which makes fixing them expensive. Also, plan-based methods have a hard time dealing with changes in requirements, which happen a lot in development projects.

On the other hand, Agile methods were designed to address these shortcomings through continuous deployment and software testing. However, because Agile focuses on detailed planning for the next iteration only, it does not facilitate long-term planning or effectively manage large projects (Schwaber & Sutherland, 2016).

To overcome the limitations of both traditional and agile methods, organisations often adopt a blended strategy known as hybrid development approaches or hybrid approaches. According to Kuhrmann et al. (2017), a hybrid approach is defined as any combination of agile methodologies with traditional development approaches, such as plan-driven development approaches, that an organisational unit adopts and customises to its own context needs, such as application domain, culture, process, project, organisational structure, techniques, or technologies.

Hybrid Agile offers a tailored solution that aligns with diverse project requirements and organisational contexts. It addresses the limitations of purely Agile or Waterfall methodologies by integrating their strengths, thereby providing a more adaptable and comprehensive framework (Bick et al., 2018). This hybrid approach enables teams to maintain a clear vision and strategic direction while fostering collaboration, innovation, and rapid response to change.

Several publications have already been written about the study of hybrid approaches. Theocharis et al. (2015) SLR examined the application of plan-based and agile methodologies. They reported a widespread application of mixed approaches for those techniques. Considering their conclusions, the HELENA study was carried out, in which practitioners were questioned regarding their approach and practice use. The study was addressed to practitioners, and questions about their methods and practice use were asked. The study was published in 2017 and resulted in 1467 data points with 691 complete answers (Kuhrmann et al., 2017). Noll and Beecham (2019) analysed the data and found that in 66 per cent of the cases the Hybrid Agile approach was used.

Prenner (2020) investigated Hybrid Agile and found that methods and frameworks like Scrum, Iterative Development, Kanban, Waterfall, and DevOps are often used to create combinations of methods. The common Hybrid Agile model is depicted in Figure 1.

Due to the extensive adoption of the Hybrid Agile method in the industry (Noll and Beecham, 2019), we argue that security in the Hybrid Agile development method deserves as much attention as the pure Agile method, and our proposed framework contributes to the implementation of Secure Software Engineering in the Hybrid Agile development process.

Secure Software Engineering (SSE)

Secure Software Engineering (SSE) integrates a set of systems engineering practices and activities into the software development process. Khan et al. (2022) did a systematic mapping study on the Security Procedures in SSE. They listed a few groups of security metrics, standards and tools being used in the SSE area.

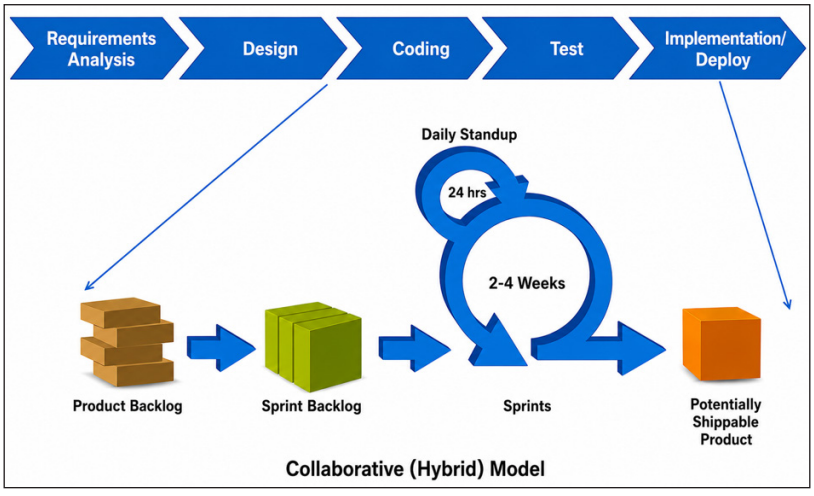

Figure 1. Hybrid agile model (Prenner, 2021)

In general, the security issues are addressed by two development models (International Organisation for Standardisation, 2008): (1) The ISO/IEC's highly process-oriented security management, metrics, and implementation framework is called the Secure Software Engineering Capability Maturity Model (SSE-CMM) (ISO/IEC, 2008). The Capability Maturity Model Integration, created by Carnegie Mellon University and now maintained by the CMMI Institute (Synopsys Software Integrity Group, 2017), is the model's source. Applying the CMM-based model as a process model can be highly expensive (Diaz et al., 2009), and its potential applications go beyond security enhancement. (2) The Software Assurance Maturity Model (SAMM), developed by the Open Web Application Security Project (OWASP) and licensed under an open-source licence, includes development-time activities as well as best practices for security governance, construction, verification, and operations. The SAMM has taken the place of the Comprehensive, Lightweight Application Security Process (CLASP), a model that was specifically developed for software development and was previously maintained by OWASP (Rindell et al., 2018). Additionally, there are clear parallels between SAMM and the Building Security in Maturity Model (BSIMM) (Synopsys Software Integrity Group, 2017). The BSIMM, an annually published survey of prevailing best practices in Secure Software Engineering, does not aim to define specific security models or frameworks. Rather, it serves as a descriptive benchmark that reflects the current state of information security practices across the industry.

The work of Kagombe et al. (2021) proposed an Agile Security framework based on the SSE-CMM's security engineering model and features three areas: security risk analysis, security engineering and assurance.

Most of the work in Secure Software Engineering (SSE) focuses on pure Agile development methodology, where many publications report on aligning the Agile

development activities with security practices (Rindell et al., 2017; Siiskonen et al., 2014; Tøndel et al., 2022; Türpe & Poller, 2017; Van der Heijden et al., 2018; Voggenreiter & Schöpp, 2023). However, for SSE implementation in Hybrid Agile development, the publications are still limited (Ding et al., 2020; Maidin & Yahya, 2024). Therefore, our work contributes to the investigation of the implementation of SSE in Hybrid Agile development.

This study distinguishes itself from earlier work in several important respects. Existing studies on SSE (e.g., Alshareef et al., 2024; Khan et al., 2022; Kagombe et al., 2021; Rindell et al., 2021; Thool & Brown, 2024) focus almost exclusively on pure Agile environments such as Scrum or Kanban and propose frameworks or guidelines that assume a fully iterative development model. They do not account for the structural duality of Hybrid Agile, namely, the coexistence of plan-driven phases (formal requirements elicitation, architectural design) with sprint-based incremental delivery. The work of Ding et al. (2020) and Maidin and Yahya (2024) comes closest to this study's concern, but both rely on conceptual analysis rather than primary empirical data. This study fills that gap by grounding its framework in interview-based evidence drawn from practitioners actively working within Hybrid Agile environments. Furthermore, unlike studies that focus solely on technical security mechanisms (e.g., vulnerability assessment tools, OWASP compliance), this study also captures the socio-technical and organisational dimensions of security adoption, including management enforcement, developer involvement, and training elements whose importance has been highlighted in the technology acceptance literature (Wang et al., 2024) but rarely examined in the SSE domain.

METHOD

The methodology adopted in this research is qualitative in nature. Qualitative analysis is a method of assessing non-numerical data to understand concepts, opinions, or experiences (Delahunt & Maguire, 2017). This type of analysis is often used to gain insights into people's behaviours, motivations, and attitudes. Unlike quantitative analysis, which focuses on numbers and statistical relationships, qualitative analysis emphasises understanding the qualities and characteristics of phenomena. Qualitative analysis involves interpreting data based on the researcher's perspective, often leading to a more subjective understanding of the data. One of the data collection methods used for qualitative analysis is an interview with the identified respondents.

Instrument Development

To address the research question and research objective, a set of questions was designed. The questions are included in an interview protocol that was specifically designed to facilitate the interview while adhering to the appropriate and standardised data collection procedures. The protocol was divided into three sections, consisting of the introduction

of the research project, followed by the demographic data of the respondents and the questions. The interview protocol was pilot tested by conducting a preliminary interview with a beginner software developer participating in hybrid agile projects. The input obtained from the pilot interview is being utilised for the actual data collection.

Data Collection

Based on the identified research questions, the data collection process was implemented using semi-structured interviews, where recruitment was done by distributing the call for participants through the network of former students at the university, together with their associated network. After doing the filtering based on the software development methodology applied in their company, 10 companies were selected as the participants in the study. All selected participants are implementing the Hybrid Agile development methodology to complete their projects.

Ten (10) experienced software developers representing a range of sectors were interviewed in person. All companies are in Kuala Lumpur, Malaysia, and range from medium to large companies. Each participant has over five years of professional experience as a programmer or software developer, demonstrating their expertise in the field. Each interview lasted anywhere from thirty minutes to an hour. Every interview was audio-recorded, and then transcriptions were made. The list of interview questions is provided in Table 1.

Since this study employs a qualitative research design, the notion of dataset balance differs from that applicable to quantitative or machine learning studies. Nonetheless, deliberate measures were taken to ensure representational balance across several dimensions. First, participant selection intentionally covered a variety of industry sectors (including finance, healthcare, retail, and technology), ensuring that the findings were not unduly shaped by the security culture or practices of any single domain. Second, both medium-sized and large companies were included to capture variation in organisational maturity and resource availability for security implementation. Third, all participants were required to have a minimum of five years of professional experience and to be actively working within a Hybrid Agile development environment, thereby reducing the risk of responses that reflect either inexperience or purely traditional or purely Agile contexts. Saturation was monitored during the thematic analysis process; the recurring themes across the ten interviews suggest that data saturation was largely achieved within this sample, though future studies with larger and more geographically diverse samples would further strengthen generalisability.

Data Analysis

The results of the interview were analysed using the six-step theme analysis framework (Delahunt & Maguire, 2017). Finding the themes and using them to answer the research

question or problem is the aim of thematic analysis. A thematic analysis provides insight into the elaborated details of the data.

Table 1
List of interview questions

	Interview Questions
1	There are many types of development models, like Waterfall, Scrum or approaches like prototype or eXtreme Programming. Could you tell me what development framework/ model/approach/technique you implement in a software project?
2	Is there any difference between the model adopted for small-scale projects and a large-scale project?
3	In a software project, a software engineering team will follow certain development phases like requirements gathering, analysis and design, coding, testing, and project deployment. Could you please explain the development phases [mention the model that the respondent mentioned in the above questions]?
4	To what extent does your Software Engineering team execute the [mention the models that the interviewee implements] in a software project?
5	Which project development phase is crucial to execute?
6	Which development phase is important, will it be design, coding or testing?
7	Do your priorities change when a deadline approaches?
8	If the answer is YES, ask this question How are the changes executed and implemented?
9	If the answer is NO, ask this question: Why are the priorities not being changed, although the deadline is approaching?
10	I am going to mention three terms. The terms are: a) Secure Software Development Lifecycle, b) Open Web Application Security Project (OWASP) and c) Threat Model Do any of those three terms sound familiar to you?
11	If the answer is YES, ask these questions:
12	Do you use the [mention the term] in your software development project?
13	How does the implementation take place?
14	Besides the approach that you mentioned, is there any other security model/ framework/ approach/ technique/ tool that you execute in your project?
15	If the answer is NO, ask this question: Could you please explain the security measures that you and your development team execute in a software project?
16	In your opinion, are the above security measures that you implement sufficient for a software project? If the answer is YES, no question to ask.
17	If the answer is NO, ask this question: Could you please suggest any methods for improvement?
18	What is your opinion on Hybrid Agile?
19	What are the Strengths?
20	What are the Weaknesses?

The gathered data is interpreted and analysed through a thematic analysis. The utilisation of the primary interview questions as themes is a common thematic analysis drawback (Delahunt & Maguire, 2017). This illustrates the fact that rather than being analysed, the data have been arranged and condensed.

Data analysis for interviews involves several steps to systematically evaluate and interpret the qualitative data collected. The goal is to derive meaningful patterns, themes, and insights from the interview transcripts or recordings.

The information obtained from the conducted interviews is provided in the next section. Participants in the study were interviewed to learn about the perspectives of software developers on secure software development and hybrid software development methodologies. The six steps are as follows: familiarise yourself with the data, create preliminary codes, look for themes, evaluate themes, define themes, and write up. It can repeatedly go between the steps and back, especially when handling complex data.

Three-stage Coding Phase

The interview is observed with a structured three-stage coding phase as follows:

Preliminary coding: Transcripts were read repeatedly to achieve a thorough understanding of the feedback. Open coding was executed at the sentence level by assigning descriptive labels to text segments. For example, comments regarding the project manager checking security rules periodically were coded under operational enforcement, while remarks detailing rushing code to meet active sprint timelines were coded as temporal boundary pressures.

Searching & Evaluating Themes: Coding was used to cluster interrelated open codes into cohesive categories based on shared structural properties. These categories mapped out specific operational, psychological, and organisational dimensions within the hybrid environment.

Defining Themes & Documentation: Selective coding abstracted these categories into our seven high-level themes. To enhance the credibility and practical relevance of these themes, they were systematically cross-referenced with and justified by established Secure Software Engineering (SSE) literature and architectural frameworks, the Open Web Application Security Project (OWASP) governance standards and the Building Security in Maturity Model (BSIMM) software security framework. The theme coding matrix and the literature grounding for the seven themes are shown in Table 2.

RESULTS DISCUSSION

The interview questions were divided into several themes to capture the different dimensions of the required information. The list of themes is given in Figure 2.

Table 2
Analytical coding matrix and literature grounding for the 7 derived themes

Derived Theme	Target Code Focus	Key Transcript Keywords	Supporting Literature / Framework Justification
Theme 1: Protocol/ Process	Evaluation of standardised security policies, compliance baselines, and formal checklists across hybrid cycles.	Policy, checklist, guidelines, standard, process	OWASP SAMM Governance & Construction Domains: Justified by industry standards evaluating how compliance mandates shape engineering behaviour.
Theme 2: Manager/ Supervisor	The role of operational leadership in monitoring compliance, administering briefings, and enforcing guidelines.	Enforce, audit, supervisor, manager, checkpoint	Security Enforcement Models (Al-Amin et al., 2018): Tracks the impact of managerial intervention and sanction policies on practice adoption.
Theme 3: Automate/ Manual Tools	The practical application and developer preference for automated testing hooks vs labour-intensive reviews.	Automated tool, plugin, scan, manual check, plugin	BSIMM SSDL Touchpoints: Maps code analysis, static scanning, and automated verification touchpoints within fast iterations (Montuban, 2024).
Theme 4: Individual Developer Priority	Developer task ownership and cognitive focus when balancing feature completion timelines against secure coding rules.	Workload, delivery, shortcut, blind spot, feature rush	Cognitive Blind Spots Framework (Oliveira et al., 2014): Analyses why rapid developmental speeds naturally displace non-functional security goals.
Theme 5: Client/User	The external pressure from clients focusing strictly on functional requirements at the cost of software resilience.	Functionality, client demand, system feature, visibility	Agile Manifesto Customer Collaboration Axis: Examines the tension between visible functional value and invisible secure logic. (Agile Alliances, 2019).
Theme 6: Training	The availability and effectiveness of ongoing structured training programs to address technical security gaps.	Security education, baseline training, capability, awareness	Microsoft SDL Core Practice 1 (Security Training): Validates the baseline training prerequisite required for engineers before sprint allocation (Microsoft Corporation, 2019).
Theme 7: Group/Team	Cross-team alignment, communication channels, and collaborative resource sharing across modular sprint units.	Cross-team, collaboration, support unit, security team	Scrum Framework Synergies (Schwaber & Sutherland, 2020): Evaluates how cross-functional and multidisciplinary teams coordinate complex system dependencies

For Theme 1, the findings show that for most of the companies, the implementation of secure programming in the hybrid development method depends on the company's policies, and there are no common practices identified among the companies involved.

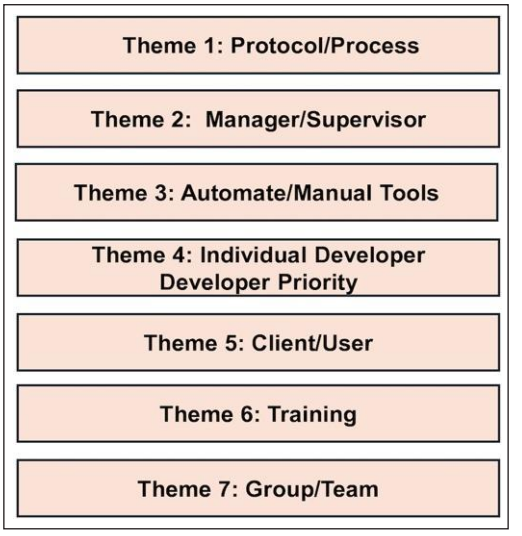

Figure 2. List of interview themes

Furthermore, for Theme 2, most of the interviewees expressed uncertainty regarding the Secure Software Engineering methods and practices that should be implemented or that have already been applied. In this context, the enforcement role of managers and supervisors becomes particularly crucial. They are responsible for ensuring that the Hybrid Agile methodology and Secure Software Engineering practices are properly implemented and adhered to, while also providing security briefings and training to junior developers.

For Theme 3, both automated and manual tools may contribute to the implementation of secure software practices in development. The developers prefer the use of automated tools, which ease their jobs.

For Theme 4, developers' active involvement in the design of security procedures and protocols is crucial, as it significantly influences the adoption and consistency of secure software development practices. When developers contribute to the creation of security methods and frameworks, they are more likely to perceive security as an integral part of their professional role and responsibility.

For Theme 5, the clients do not see security as important; they are mainly concerned about the functionality of the system.

For Theme 6, effective training reinforces security practices, standards, and policies, while also aligning software security requirements with insights derived from emerging technical capabilities and newly available data.

Lastly, for Theme 7, the participants consistently highlighted that cross-team collaboration is vital to the success of secure software development. They acknowledged receiving support from development peers, specialised units such as support, network, and security teams, and strong backing from top management.

In one question, the participants were asked about the Security Model being used in their company, and their responses are shown in Table 3.

Most of the companies use the Open Web Application Security Project (OWASP), and half of the participants responded that they use a variety of methods, including code security review, penetration testing, vulnerability assessment, and security monitoring. and security requirement specifications.

The participants were asked, "How does the implementation take place?" We found that most of the companies have security checklists for testing procedures and reporting. Several developers reported that the managers would go through the checklist periodically to ensure that the proper security practices were adhered to by the team members.

In addition, we asked the question about other additional security model/frameworks/approaches/techniques/tools that they use in their project. The answers are shown in Table 4.

The results obtained from the study support our hypothesis that Secure Software Engineering practices are implemented in the Hybrid Agile process.

Table 3
Security models used in company

Security Model	#
Secure Software Development Lifecycle	1
Open Web Application Security Project (OWASP)	4
Threat Model	0
Others	5

Table 4
Other security practices

#	Answers
1	Code Security Review
2	Penetration Testing
3	Vulnerability Assessment
4	Security Monitoring
5	Firewall
6	Intrusion Detection Systems
7	Access Control List

Secure Hybrid Agile Framework

Based on the answers from the respondents in the interview, we have developed the Secure Hybrid Agile Framework, as shown in Figure 3.

The development process begins with the planning phase, which is overseen by a project manager. During this phase, a kick-off meeting is held to discuss the project, and tasks are divided by modules. This phase also involves initial communication and agreement with the client to ensure alignment on objectives and expectations.

Following the planning phase is the estimation phase. During this stage, the team decides what the new product will be, acquires requirements, and decides which tasks are most important. This makes sure that everyone understands the project's goals and that tasks are grouped by how important and urgent they are. At this point, the planning for the security implementation will happen, and, later, it will be translated into the design and development stage of the software development.

The high-level design or architecture phase is the last phase in this early stage of development. In this phase, the team works on the system's high-level design or architecture.

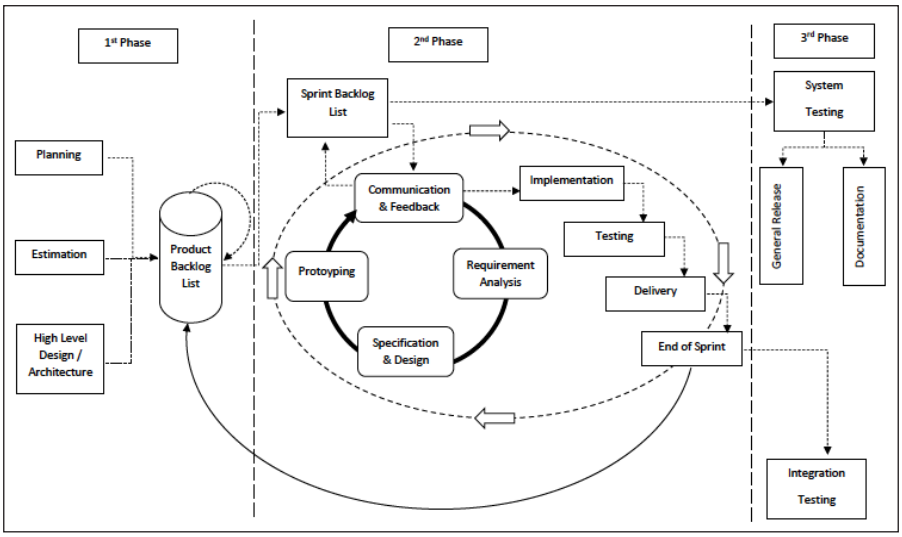

Figure 3. Secure hybrid agile framework

They find the quality traits and units of measurement that the design needs to meet certain standards. They also work on User Interface (UI) and User Experience (UX) design, implementing the requirements from the product backlog list to create an intuitive and effective user experience.

In the middle stage, the development work is broken down into sprints. The work is carried out iteratively and then sequentially. In the sprint, there are activities done iteratively, including communication, requirement analysis, specification, design, and prototyping activities, until the customer or client comes to a consensus with the developer. In the sprint stage, there are sprint planning meetings, daily scrum, peer review, effort and cost update, and addition or removal of the enhancement depending on the need.

After the agreement by the customer or client, the rest of the development work, including implementation, testing, and the delivery of the new increment, takes place in an ordered manner, like the Waterfall model approach. In fact, this stage covers front-end and back-end programming to implement the new features. At the end of the sprint cycle, the newly produced increment will then be integrated with the existing system that already exists. The end of this phase will then see the team hold a sprint review meeting, where the team will review the work completed and plan the next sprint.

The third phase is the final phase, where integration testing is conducted after the completion of each sprint. The main objective is to make sure that the new pieces of functionality integrate perfectly with the existing application. System testing comes next after integration testing is finished. This includes careful testing and release procedures to verify the stability and usability of the product. Additionally, security testing is done to find possible weaknesses and holes in the system. The product is prepared for widespread

release following system testing and after the required updates and changes have been made to the documentation.

Lessons Learned

Generally, Secure Security Engineering in the Hybrid Agile approach research area still needs further exploration (Ding et al., 2020; Maidin & Yahya, 2024). To increase the software's security, more research is required to determine how well security integration techniques work in the development process. The Secure Hybrid Agile framework's suggested methodology places a strong emphasis on incorporating security elements into software development and planning during the initial stages of the software lifecycle (Gill et al., 2018; Imani & Nakano, 2018; Prenner et al., 2021).

According to the interview, one of the factors contributing to the integration of security into the development lifecycle is security training. The developers stated that gaining knowledge about security-related issues in systems enhances their understanding of the significance of security components in the system's development process. This aligns with the results indicated by Rindell et al. (2018), where the Agile developers gain knowledge from the various security training provided before the start of the project.

Another important motivation for security inclusion in the development lifecycle is the enforcement of security policies by the project manager to ensure that secure software practices are followed and executed accordingly. This finding was also highlighted by Tøndel et al. (2022) and Voggenreiter and Schöpp (2023), where the security champions in the organisation should enforce the security guidelines and communicate clearly to the project members.

Threats to validity

Generally, the results from the interview have their own threats to validity.

- Internal validity: The interviews were conducted by 2 interviewers. Therefore, the interview style may contribute to the participants' responses to the questions. We tried to minimise the threat by preparing a common transcript for the interview.
- External validity: The number of respondents is limited (10). This could mean the results cannot be generalised outside the research environment. However, all respondents are experienced developers; thus, their feedback is valuable regarding this topic.

In addition to the threats above, several further limitations of this study must be acknowledged. First, the sample is restricted to ten companies, all based in Kuala Lumpur, Malaysia. While this geographic focus provides contextual coherence, it limits the transferability of findings to organisations in other national or cultural contexts where software development practices, security regulation, and organisational norms may differ

substantially. Second, the qualitative, interview-based design, although well-suited to exploratory inquiry, means that the findings cannot be statistically generalised to the broader population of Hybrid Agile organisations. Third, the study relies on self-reported perceptions of security practice; there is no independent verification of whether the stated practices were implemented as described, which introduces potential social desirability bias.

To mitigate this external validity limitation and strengthen the credibility, generalizability, and practical relevance of the study, a domain expert validation process was introduced. Rather than relying on a single perspective, the proposed Secure Hybrid Agile Framework was evaluated by an independent panel of domain experts, software engineers with over 10 years of experience managing large-scale hybrid architectures. These experts evaluated the framework for process disruption, sprint velocity impacts, and developer workflow feasibility.

CONCLUSION

In summary, this research work deals with the highlighted integration of Secure Software Engineering processes within Hybrid Agile methodologies. The findings of this research work reveal gaps in security processes and stress the key roles of enforcement by management, participation and support by developers, automation tools, and overall training for improving secure software development in Hybrid Agile settings.

The Secure Hybrid Agile Framework provides a structured approach for incorporating security at every stage of the Hybrid Agile phase, which aims at ensuring that security is taken into consideration at every stage of development, right from planning to deployment. The framework mentioned above has its own objective of making sure that vulnerabilities are removed right from the start, during the software development stage.

By offering even more insight into the opportunities and challenges posed by the Hybrid Agile method of software development, this work further improves the security and resilience of the software development environment. Future research should validate and refine the proposed framework to ensure it remains pliable and responsive to the rapidly changing environment presented by software development security. Future work can involve extending this study to a wider audience and examining security behaviours presented by the participants.

To effectively address the empirical evidence demonstrating the efficacy of our framework, future research will be conducted to prove the framework within actual software projects.

ACKNOWLEDGEMENT

This research is fully funded by the Fundamental Research Grant Scheme, grant number FRGS/1/2019/ICT01/UIAM/03/2.

REFERENCES

Agile Alliances. (2019). *12 principles behind the Agile Manifesto.*

Al-Amin, S., Ajmeri, N., Du, H., Berglund, E. Z., & Singh, M. P. (2018). Toward effective adoption of secure software development practices. *Simulation Modelling Practice and Theory, 85*, 33-46.

Alshareef, R., Alshabeeb, E., Alakkas, N., & Niazi, M. (2024). Challenges in developing secure software within agile environments. In *Proceedings of the 28th International Conference on Evaluation and Assessment in Software Engineering (EASE 2024)* (pp. 1-10). Association for Computing Machinery. https://doi.org/10.1145/3661167.3661284

Asthana, V., Tarandach, I., O'Donoghue, N., Sullivan, B., & Saario, M. (2012). *Practical security stories and security tasks for agile development environments.* SAFECode.

Bick, S., Spohrer, K., Hoda, R., Scheerer, A., & Heinzl, A. (2018). Coordination challenges in large-scale software development: A case study of planning misalignment in hybrid settings. *IEEE Transactions on Software Engineering, 44*(10), 932-950. https://doi.org/10.1109/TSE.2017.2730870

Delahunt, B., & Maguire, M. (2017). Doing a thematic analysis: A practical, step-by-step guide for learning and teaching scholars. *All Ireland Journal of Teaching and Learning in Higher Education, 8*(3), 3351-3359. https://doi.org/10.5281/zenodo.5828312

Ding, C. B., Maidin, S. S., Medi, I., & Nghiem, T. L. (2020). Secure software implementation in hybrid agile development approach. *International Journal of Management, 11*(10), 1713-1721. https://doi.org/10.34218/IJM.11.2020.157

Gill, A. Q., Henderson-Sellers, B., & Niazi, M. (2018). Scaling for agility: A reference model for hybrid traditional-agile software development methodologies. *Information Systems Frontiers, 20*(2), 315-341. https://doi.org/10.1007/s10796-016-9672-8

Imani, T., & Nakano, M. (2018). A model for effective area of hybrid approach combining agile and plan-driven methods in IT projects. In *Proceedings of the International Association for P2M, Spring* (pp. 49-64). https://doi.org/10.20702/iappmproc.2018.Spring.0_49

International Organisation for Standardisation. (2008). *Information technology-Security Techniques-Systems security engineering -Capability maturity model (SSE-CMM) (ISO/IEC 21827).* https://doi.org/10.3403/30143286

Kagombe, G. G., Mwangi, R. W., & Wafula, J. M. (2021). Achieving standard software security in agile developments. In *Proceedings of the 11th International Conference on Information and Communication Management (ICICM '21)* (pp. 24-33). Association for Computing Machinery. https://doi.org/10.1145/3484399.3484403

Khan, R. A., Khan, S. U., & Ilyas, M. (2022). Exploring security procedures in secure software engineering: A systematic mapping study. In *Proceedings of the 26th International Conference on Evaluation and Assessment in Software Engineering (EASE '22)* (pp. 433-439). Association for Computing Machinery. https://doi.org/10.1145/3530019.3531336

Kuhrmann, M., et al. (2017). Hybrid software and system development in practice: Waterfall, Scrum, and beyond. In *Proceedings of the 2017 International Conference on Software and System Processes (ICSSP 2017)* (pp. 30-39). Association for Computing Machinery. https://doi.org/10.1145/3084100.3084104

Maidin, S. S., & Yahya, N. (2024). An insight into hybrid agile software development approach and software security among software engineers: A critical evaluation. *Journal of Theoretical and Applied Information Technology*, 450-459.

Maier, P., Ma, Z., & Bloem, R. (2017). Towards a secure SCRUM process for agile web application development. In *Proceedings of the 12th International Conference on Availability, Reliability and Security (ARES '17)*. Association for Computing Machinery. https://doi.org/10.1145/3098954.3103171

Microsoft Corporation. (2019). *Microsoft Security Development Lifecycle*. Microsoft. https://www.microsoft.com/en-us/securityengineering/sdl

Montuban, N. (2024). *BSIMM (Building Security in Maturity Model): A complete guide.* Codific. https://codific.com/bsimm-building-security-in-maturity-model-a-complete-guide/

Noll, J., & Beecham, S. (2019). How agile is hybrid agile? An analysis of the HELENA data. In *PROFES 2019* (pp. 341–349). Springer. https://doi.org/10.1007/978-3-030-35333-9_25

Oliveira, D., Rosenthal, M., Morin, N., Yeh, K. C., Cappos, J., & Zhuang, Y. (2014). It's stupid psychology: How heuristics explain software vulnerabilities and how priming can illuminate developers' blind spots. In *Proceedings of the 30th Annual Computer Security Applications Conference* (pp. 296-305).

Prenner, N. (2020). Towards improving the organisation of hybrid development approaches. In *Proceedings of the International Conference on Software and System Processes (ICSSP '20)* (pp. 185-188). Association for Computing Machinery. https://doi.org/10.1145/3379177.3390304

Prenner, N., Unger-Windeler, C., & Schneider, K. (2020). How are hybrid development approaches organised? A systematic literature review. In *Proceedings of the International Conference on Software and System Processes (ICSSP '20)* (pp. 145-154). Association for Computing Machinery. https://doi.org/10.1145/3379177.3388907

Prenner, N., Unger-Windeler, C., & Schneider, K. (2021). Goals and challenges in hybrid software development approaches. *Journal of Software: Evolution and Process, 33*(11), Article e2382. https://doi.org/10.1002/smr.2382

Rindell, K., Hyrynsalmi, S., & Leppänen, V. (2017). Busting a myth: Review of agile security engineering methods. In *Proceedings of the 12th International Conference on Availability, Reliability and Security (ARES '17)* (pp. 74:1-74:10). Association for Computing Machinery. https://doi.org/10.1145/3098954.3103170

Rindell, K., Hyrynsalmi, S., & Leppänen, V. (2018). Aligning security objectives with agile software development. In *Proceedings of the 19th International Conference on Agile Software Development: Companion (XP '18)* (pp. 3:1-3:9). Association for Computing Machinery. https://doi.org/10.1145/3234152.3234187

Rindell, K., Ruohonen, J., Holvitie, J., Hyrynsalmi, S., & Leppänen, V. (2021). Security in agile software development: A practitioner survey. *Information and Software Technology, 131*, Article 106488. https://doi.org/10.1016/j.infsof.2020.106488

Schwaber, K., & Sutherland, J. (2020). *The Scrum guide*. https://www.scrum.org/resources/scrum-guide

Siiskonen, T., Sars, C., Vähä-Sipilä, A., & Pietikäinen, A. (2014). Generic security user stories. In P. Pietikäinen & J. Röning (Eds.), *Handbook of the secure agile software development life cycle* (pp. 9-14). University of Oulu.

Synopsys Software Integrity Group. (2017). *The building security maturity model*. Synopsys.

Tell, P., Klünder, J., Küpper, S., Raffo, D., MacDonell, S. G., Münch, J., Pfahl, D., Linssen, O., & Kuhrmann, M. (2019). What are hybrid development methods made of? An evidence-based characterisation. In *Proceedings of the International Conference on Software and System Processes (ICSSP '19)* (pp. 105-114). IEEE. https://doi.org/10.1109/ICSSP.2019.00022

Theocharis, G., Kuhrmann, M., Münch, J., & Diebold, P. (2015). Is Water Scrum-Fall reality? On the use of agile and traditional development practices. In *Product-focused software process improvement* (pp. 149-166). Springer. https://doi.org/10.1007/978-3-319-26844-6_11

Thool, A., & Brown, C. (2024). Securing agile: Assessing the impact of security activities on agile development. In *Proceedings of the 28th International Conference on Evaluation and Assessment in Software Engineering (EASE 2024)* (pp. 1-11). Association for Computing Machinery. https://doi.org/10.1145/3661167.3661280

Tøndel, I. A., Cruzes, D. S., Jaatun, M. G., & Sindre, G. (2022). Influencing the security prioritisation of an agile software development project. *Computers & Security, 118*, Article 102744. https://doi.org/10.1016/j.cose.2022.102744

Türpe, S., & Poller, A. (2017). Managing security work in Scrum: Tensions and challenges. In *Proceedings of the International Workshop on Secure Software Engineering in DevOps and Agile Development (SecSE 2017)* (pp. 34-49). CEUR Workshop Proceedings.

Van der Heijden, A., Broasca, C., & Serebrenik, A. (2018). An empirical perspective on security challenges in large-scale agile software development. In *Proceedings of the 12th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM '18)* (Article 45). Association for Computing Machinery. https://doi.org/10.1145/3239235.3267426

Voggenreiter, M., & Schöpp, U. (2023). Prioritising industrial security findings in agile software development projects. In *Proceedings of the 45th International Conference on Software Engineering: Companion Proceedings (ICSE '23)* (pp. 375-379). IEEE Press. https://doi.org/10.1109/ICSE-Companion58688.2023.00106

Wang, Y., Hung, J. C., Huan, C. H., Hussain, S., Yen, N., & Jin, Q. (2024). Design of TAM-based framework for credibility and trend analysis in sharing economy: Behavioural intention and user experience on Airbnb as an instance. *Computer Science and Information Systems, 21*(2), 547-568. https://doi.org/10.2298/CSIS230323010W