Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

AI-Augmented Agile MVP Development for Lean Tech Startups

Muhammad Owais Mushtaq

k248335@nu.edu.pk

National University of Computer and Emerging Sciences

Research Article

Keywords: Agile, Minimum Viable Product, Design Science Research, Generative AI, Lean Tech Startups, Verification Fatigue

Posted Date: September 3rd, 2026

DOI: https://doi.org/10.21203/rs.3.rs-10346557/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: No competing interests reported.

AI-Augmented Agile MVP Development for Lean Tech Startups

Muhammad Owais Mushtaq
Department of Computer Science
National University of Computer & Emerging Sciences, Karachi, Pakistan
k248335@nu.edu.pk

May 2026

Abstract

Lean tech startups face many challenges to timely deliver Minimum Viable Products (MVPs) despite working in an agile environment and applying lean principles. This delayed delivery can cause product failure in terms of poor product quality or the delayed delivery itself, and sometimes even venture closure too (because some startups completely rely on one product). Most of the time, the reason behind this problem is a tight budget and a small, inefficient engineering team with scarce resources. Using Design Science Research (DSR) methodology, this research included and evaluated semi-structured interviews with 18 industry practitioners to understand the problem and then proposed an AI-augmented agile MVP development framework. As per the framework, AI-assisted workflows were applied to a live enterprise setup during a legacy API migration project in two phases: inception and planning, followed by implementation and quality assurance. During the first phase, the framework reduced user story estimation time by 50% while maintaining a 94% accuracy rate compared to domain experts. In the second phase, AI-assisted development achieved up to 97% migration accuracy with minimal iteration, while AI-supported code reviews and testing improved software quality and streamlined quality assurance activities. Ultimately, this research demonstrates that AI's effective integration can overcome the challenges of small, inefficient engineering teams. The proposed framework will help lean tech startups adopt a workable way to adopt AI-first agile workflows that increase development speed without compromising quality.

Keywords: Agile, Minimum Viable Product, Design Science Research, Generative AI, Lean Tech Startups, Verification Fatigue.

Acknowledgments

I am extremely thankful to my Supervisor, Mr. Ubaid Aftab, for his excellent guidance, insightful feedback, technical expertise, and support in completing this research. Furthermore, I would like to acknowledge the support rendered by my co-supervisor, Dr. Nazia Nazeer, in form of her constructive comments and guidance, which helped tremendously in formulating the research methodology used in this thesis.

Additionally, I would like to express my gratitude to the faculty and administrative staff of MS Software Project Management (MS SPM) department of National University of Computer & Emerging Sciences (FAST-NUCES), Karachi for providing me with the required environment for research.

Finally, I am extremely thankful to the management and fellow colleagues at Matrix Systems (Pvt.) Ltd. for providing the required infrastructure and professional environment to conduct this research. Evaluation of this research in an actual corporate environment played a vital role in its successful completion.

Abbreviation Notation and Nomenclature

AC	Acceptance Criteria
AI	Artificial Intelligence
API	Application Programming Interface
CI/CD	Continuous Integration / Continuous Deployment
CMMI	Capability Maturity Model Integration
CRUD	Create, Read, Update, Delete
DSR	Design Science Research
E2E	End-to-End
GenAI	Generative Artificial Intelligence
IDE	Integrated Development Environment
JPA	Java Persistence API
KPI	Key Performance Indicator
LLM	Large Language Model
LOC	Lines of Code
MS	Master of Science
MVP	Minimum Viable Product
PM	Project Manager
PO	Product Owner
PR	Pull Request
QA	Quality Assurance
SDE	Software Development Engineer
SDLC	Software Development Life Cycle
SPM	Software Project Management
SQAE	Software Quality Assurance Engineer

1 Introduction

The software development industry is rapidly growing into a new era driven by Generative Artificial Intelligence (GenAI), especially for fast development. The pace and quality of development are crucial for small-scale startups, particularly when developing Minimum Viable Products (MVPs), as an MVP establishes the product's entire lifespan and whether it will be continued or dropped. This chapter outlines the background, problem statement, motivation, and the main research questions that this thesis will try to answer.

1.1 Background

Most tech startups and enterprises face a critical bottleneck when rapid development is required under tight deadlines. One of the most important examples of this is the development of an MVP. Past research demonstrates that a proper framework for MVP development can accelerate product development in an engineering context [1]. Although Agile methodologies promote iterative development and delivery, all stages of the SDLC remain completely manual and highly dependent on humans, whether it is planning, estimation, backlog refinement, or mainly development and testing. Recent studies on GenAI adoption in startups highlight its potential to improve efficiency and support decision-making across workflows [2]. Slow development iterations remain a common reason for startup failure, particularly when technical execution fails to align with strategic objectives.

While GenAI has reshaped many industries and significantly improved workforce productivity on a global scale [3], its major impact on standard software project management processes, as outlined in the PMBOK, introduces new opportunities alongside integration challenges [4]. To face these challenges, from planning to development, requires a very structured and disciplined approach to ensure effective adoption. Moreover, traditional mindsets surrounding AI-driven startups or organizational models must evolve to accommodate faster, data-driven development paradigms [5].

1.2 Problem Statement

The majority of startups use Agile and Lean methodologies, and yet they still fail early in their lifecycles, not because they compromise on technology, nor do their teams lack skills and expertise; rather, it is because they do not analyze requirements properly and cannot build a proper understanding of what clients really want and what the actual market demand is [6]. Additionally, most startups' Agile planning depends heavily on subjective prioritization models like MoSCoW, which fail to account for technical risks and effort, ultimately leading to misaligned MVP features [7].

While GenAI tools (such as Claude, GitHub Copilot, Cursor, and Gemini) exist to speed-up the coding, project managers and developers still lack an empirically validated framework to integrate these tools into an end-to-end Agile lifecycle that tackles these specific failure points. Currently, most workflows are not designed for AI and do not formally welcome it into their processes. AI is usually used for short, isolated tasks like small chat prompts, but properly integrating and operating these tools still requires a dedicated framework [8]. An AI-enhanced approach is necessary for the transition from subjective manual estimation to data-driven, objective MVP development.

1.3 Motivation

Small startups that work with limited resources such as money, time, or human resources can take advantage of using GenAI's rapid growth potential and compete more effectively with bigger, stable companies [9]. Motivated by these emerging opportunities, this study investigates the following:

- **Automated Estimation:** In Agile software development environments, using GenAI-based models to support story point estimation to reduce planning overhead in the inception phase of the project [10].
- **Impact of MVP Development:** To understand how to build MVP faster and the impact of this rapid development on the overall product's growth, maintenance, and quality [11].
- **Workforce Agility and Quality:** AI is not intended to take the role of human developers because it assists developers by providing helpful recommendations, reducing their mental burden, and assisting them in making better choices. AI can speed up their work and help them pick up new skills more rapidly; this is especially helpful for developers with less expertise [12, 13]. To avoid being overconfident or placing too much reliance on the AI, developers must nevertheless thoroughly verify the accuracy and dependability of AI-generated outcomes [14].

- **Sustainable Decision-Making:** AI should not only boost the developers' ability to write code faster. It should also help startups make smarter product decisions early in the project inception phase, so they can build products that are useful, successful, and profitable [15].
- **Organizational Adaptability:** Understanding that successful AI-driven change within today's software houses depends on organizational agility and flexibility [15].

1.4 Research Questions

To close these gaps, this study addresses the following research questions:

1. How can a Design Science Research (DSR) methodology successfully integrate Generative AI agents into different phases of Agile software development, such as inception (planning and estimation) and execution (implementation and testing)?
2. In an real-life example of a legacy framework migration project, how accurate and efficient are AI-generated story point estimates compared to human expert estimates?
3. What is the impact of AI augmentation on defect density, developer flow, and verification fatigue, and how much does it reduce the time to market for MVP development using tools like Claude, Copilot, or Cursor?

1.5 Research Methodology

This thesis uses the Design Science Research (DSR) methodology to effectively answer the above research questions. DSR is a good fit for this research since it emphasizes creating an inventive artifact, to solve a specific, real-world problem using its three major cycles: relevance, design, and rigor. The framework this research has produces via DSR is the "AI-Augmented Agile MVP Development Framework." These three cycles of DSR were executed in this research as follows:

- **Relevance Cycle:** This stage highlights practical issues with the MVP of legacy code/framework migration from inception to implementation. To make sure the issue was founded in reality, data was gathered through semi-structured interviews with 18 industry professionals.
- **Design Cycle:** The proposed framework was created during this stage, which caters to both the Inception and Implementation phases of the project.
- **Rigor Cycle:** This cycle of methodology evaluated the effectiveness of the proposed framework in two phases on a live real-world migration of a complex MVP, focusing on performance, estimation accuracy, and development efficiency.

Further details on experimental setup, data collection methods, and evaluation procedure are provided in Section 3.

1.6 Dissertation Contributions

Contributions made by this study include:

1. **The AI-Augmented Agile MVP Development Framework:** This study develops and evaluates an end-to-end Design Science Research framework that incorporates multi-agent AI tools during the planning and development stages.
2. **Empirical Validation of AI Estimation:** The research provides quantitative proof that combining GenAI tools can easily shorten the time for estimation from legacy code metadata while maintaining the accuracy of estimation compared to a domain expert.
3. **AI Tool Performance and QA Integration:** This study compares AI coding assistants such as Cursor, Claude, and GitHub Copilot to get the most effective one during development and also sees the utilization of ChatGPT and Gemini in Quality Assurance workflows.
4. **Identification of Verification Fatigue:** AI saves a lot of developer time by helping them write code more efficiently and faster; human effort is still required to validate the output to minimize hallucinations and wrong requirement interpretations. This highlights the significance of human oversight in AI-assisted development and presents the idea of "verification fatigue."

1.7 Structure of the Article

The rest of this article is organized as follows:

- **Section 2 (Related Work):** Critically studies the current role of generative AI in software project management, software development, Lean Startup practices, and challenges related to AI-assisted code quality and correctness.
- **Section 3 (Research Methodology):** Defines the Design Science Research framework and the live experimental setup for a legacy framework migration MVP and the specific quantitative and qualitative metrics used for evaluation.
- **Section 4 (Proposed Work):** Presents the design of the proposed AI-Augmented Agile MVP Development Framework and explains the architecture, components, and workflow of the proposed solution developed through the DSR approach.
- **Section 5 (Results & Discussion):** Finalizes and evaluates the statistical findings from using the suggested framework in a practical project. It also elaborates how frameworks resulted in estimation accuracy, development productivity, and software quality.
- **Section 6 (Conclusions & Future Work):** Summarizes the main conclusions, explains how the study addresses the research goals, discusses the study's limitations, and makes recommendations for further research.

1.8 Summary

The need for an integrated, AI-driven approach to Agile MVP development was emphasized in this section. It was useful in determining the difficulties and delays associated with manual estimating and legacy system migrations. The next section reviews the literature on Lean Startup methodologies and the use of generative AI in software engineering.

2 Related Work

2.1 Theoretical Foundations: Lean Startup and Growth Hacking

The concept of the Minimum Viable Product (MVP) is foundational to lean tech startups. A strict theoretical differentiation between simple prototypes and MVPs is crucial for validating business models as shown in [16]. AI is rapidly becoming an active catalyst in this for integration into the Lean Startup workflows that significantly accelerate the work of product innovations [17]. Additionally, AI-driven "predictions" added to the Build-Measure-Learn loop allowed for quicker assumption validation than with conventional lean techniques, which enhanced decision-making under uncertainty [18].

Under pressure to grow fast, such an acceleration takes the form of "Growth Hacking." Generative AI is increasingly applied across product, sales, and operations to facilitate fast scaling of startups [2]. The use of AI processes can drastically reduce time-to-market in creating digital products because AI can identify scaling opportunities faster than any traditional analytics process [19]. Moreover, digitalization acts as the critical means through which startups can convert their entrepreneurial learning into value creation [20]. Such acceleration facilitates flexibility and innovation in traditional industries and legacy software, too [21]. Nevertheless, such acceleration comes at a cost since structural and organizational limitations can impede the potential of using AI in this context [22]. In summary, such tools' efficacy depends crucially on the organization's data-driven culture [23]. Moreover, data-driven organizations must translate their raw data into valuable insights for ambidextrous digital transformation [24].

While Build-Measure-Learn may offer theoretical advantages, empirical evidence shows a very high death rate in early digital startups. One of the major reasons behind the aforementioned phenomenon lies in the competency gaps in the founding teams of these projects. Startup failure tends to be directly linked with the gaps in competencies like "Information-seeking" and "Customer service orientation" where overly motivated teams forego data collection and dive headfirst into development without adequate planning [6]. Moreover, to ensure survival in an uncertain environment, especially in developing economies, one must move from theory to addressing concrete functional factors such as optimal timing and identifying customers' needs [15].

2.2 Generative AI in the Inception and Planning Stage

The Inception stage, in which the definition of requirements and initial planning take place, is among the most important and difficult phases in Agile development. Generative AI systems have proven to be promising in assisting the inception process through their ability to help convert nascent ideas into concrete and actionable requirements for development [25]. After defining requirements, estimation becomes a vital task for successful planning. Multimodal generative AI systems have proven to offer more accurate estimation of story points when compared to conventional manual techniques [26]. Likewise, generative AI systems have proven capable of offering better planning efficiency by facilitating velocity estimation [10].

One important problem in this phase of development is the subjectivity of conventional requirement priority models, which typically fall apart when faced with the harsh environment of scarce resources in a startup. To address this problem, mathematical frameworks such as RECTV (Risk, Effort, Confidence, Time to Value, and Value) have been recommended as part of an attempt to bring objective mathematical analysis into agile processes [7]. The combination of Generative AI with frameworks like RECTV would allow startups to analyze historical data and market information about "Time to Value" and related risk factors of potential MVP features [27].

2.3 AI-Augmented Development and Agentic Refactoring

From planning to execution is where we come across many challenges when it comes to making sure of proper software implementation. Whereas in earlier years the generation of AI like GitHub Copilot was limited to only coding assistance, now there have been advancements in generating AI, which not only provides deep contextual understanding but helps in agentic workflows for engineering purposes.

It has been seen that the workflow based on agents within IDE, such as the workflows within Cursor and Claude-based coding assistants, offer superior performance as far as dealing with multiple files and reducing refactoring issues of complex enterprise systems, when compared to chat-based or completion-based systems like GitHub Copilot [28]. This study is in line with these experiments conducted on enterprise Java migrations.

In order to ensure the structural and reliable nature of refactoring, multi-agent LLM-based solutions have been developed. The RefactorGPT paper shows how iterative multi-agent processes result in more stable improvements in the modularity and maintainability of code than monolithic prompting methods [29]. In a similar way, prompt-based development methods like Claude Code prove how structured generation processes can keep architectural consistency during automatic code transformations [30].

2.4 Code Correctness, Security, and Quality Assurance

Whereas the adoption of generative AI in the software development process speeds up the work, there is a tradeoff between the speed and quality of coding as far as generative AI impacts the quality of code. According to studies performed with respect to GitHub Copilot, even though the tool is highly efficient at producing boilerplate code, it might generate code that is logically incorrect in some sophisticated software engineering tasks [31].

From the standpoint of security, empiric analysis demonstrates that there is no clear evidence of increased vulnerability of code generated by GitHub Copilot compared to humans; however, the behavior of GitHub Copilot differs depending on the complexity of a task [32]. At the same time, developer's actions are crucial as an excessive reliance on suggestions made by AI often results in the development of erroneous code since there will be no thorough checking of code for vulnerabilities [33, 34].

Regarding QA, conversational AI, including ChatGPT and Gemini, has been found to be very effective for test case generation and edge-case detection, improving the efficiency of QA in well-structured development settings (as seen from the experimental study of this research). Moreover, automatic code review recommendations proposed by GitHub Copilot often need substantial overriding by developers because of the lack of architectural knowledge.

To overcome these challenges, test automation frameworks, based on large language models, including generate-and-repair frameworks, have been proven capable of autonomously detecting and repairing failed test cases, outperforming static analysis methods [35].

2.5 Productivity, Developer Flow, and Workforce Agility

The measurement of the effects of artificial intelligence on the software development productivity would need going beyond metrics such as LOCs. Empirical studies have shown that developers utilizing AI tools can accomplish their tasks at a much higher rate than their peers using traditional tools [36]. Yet, the difference arises in the decrease of repetitive tasks rather than an increase in the amount of code written.

Despite all the advantages, AI-driven software development imposes new mental burdens on developers. Research indicates that there are instances in which developers face more cognitive overload as they try to examine

sophisticated products developed by AI algorithms [37]. Furthermore, empirical studies have shown that expectations regarding the efficacy of AI tools often fade in practical use cases [38].

The use of AI in software development requires proper structuring of human and AI cooperation. The study has shown that the implementation of iterative Agile development processes supported by human-AI cooperation can help prevent requirement drift and align the results produced by the system to the changing requirements [39].

Managerially speaking, successful utilization of AI solutions rests upon the ability of the management to structure its workflows accordingly. Technical leadership strategies based on deliberate design of workflows and careful AI system integration may increase the effectiveness of this process [40].

2.6 Global Market Experiences and AI Adoption

Challenges seen in lean startups also emerge in large-scale software development across the globe. From empirical evidence, it is clear that integrating AI technology into the processes requires reorientation in terms of expectations and practices of developers [38]. Once the organization adopts AI technology, the process of collaboration changes from being implementation-oriented to becoming more architecturally-oriented.

Through the integration of AI-assisted technologies, some low-level development disturbances can be addressed, such as automating routine coding practice and freeing up time for high-level design discussions and analysis [41]. Nonetheless, such an approach would also entail changing the role of developers and engaging in more cognitively challenging activities [40].

Additionally, the alignment of user requirements with AI-generated results is another major issue in an agile environment. Methods like Behavior-Driven Development (BDD) play a critical role in aligning AI-generated results with business logic and requirements [39].

2.7 Gap Analysis

While there are many studies carried out by researchers on various individual areas of the application of generative AI in software engineering, including code generation, estimation, and refactoring, there has not been enough research conducted on comprehensive solutions which will link AI-based Agile planning to AI-enabled agile implementation in practical legacy migration cases.

Specifically, very little has been done in terms of conducting empirical research on linking the use of AI in estimation, multi-agent-based development processes, and defect prevention approaches all in one Design Science Research framework. This is because most current research only addresses parts of the problem.

3 Research Methodology

3.1 Research Design: Design Science Research

The current study utilizes the Design Science Research (DSR) approach. [42] offers guidance on using DSR to complete industry-based master's theses, pointing out that the artifacts need to be designed and tested in real-life work situations with case studies being used in evaluations.

The main artifact in this research is the AI-Augmented Agile MVP Development Framework that uses generative AI technology in Agile planning, developing, and quality management processes. Thus, by considering the reasons for which these technologies have been employed in the given processes, the AI-Augmented Agile MVP Development Framework becomes the source of prescriptive knowledge for lean startups [43].

The DSR methodology comprises the following three iterative cycles:

1. **Relevance Cycle:** Issues connected with the manual estimation of the Agile approach and migration of legacy systems based on semi-structured interviews with 18 experts from the field.

2. **Design Cycle:** A framework is built following the key elements outlined in the MAISTRO methodology [44]. This framework involves an AI-assisted estimation agent, code assistants integrated into IDE (GitHub Copilot, Cursor, and Claude), as well as LLM-powered reasoning agents (Gemini and ChatGPT). Iterative improvement of the artifact is done based on feedback obtained during actual sprint runs.

3. **Rigor Cycle:** The empirical evaluation process involves the assessment of the performance of the FinTech migration project and comparison with the baseline Agile process without AI support.

3.2 Experimental Setup

Experimentation took place in an industrial environment where an organization wanted to complete an MVP of framework migration in a tight deadline with limited resources.

- **Project Scope:** Upgrade of existing architecture from Jersey to Spring Boot.
- **Phase I (Project Inception):** AI-based estimation using tools Gemini and Junie, followed by verification of results by human domain experts who set up a benchmark in order to validate estimate-effort accuracy.
- **Phase II (Development & Testing):** Evaluation of software development process in case of agentic workflow with the following allocation of AI assistants to specific tasks:
  - Usage of Cursor and Claude for performing heavy context-dependent task such as code refactoring and writing code in several files.
  - Using of GitHub Copilot for performing IDE-assisted task of writing standard code in regular code development.
  - Performing QA by generating test cases and identifying edge cases with help of ChatGPT and Gemini and conducting automatic code reviews by technical lead using assistance in code review tools.
  - Copilot code review tools were evaluated by technical leads for auto code reviews.

This tool allocation provides a framework for assessing the effectiveness of various AI-based tools in different roles while eliminating potential biases in tool performance evaluations. Additionally, the documentation of such structured tool usage becomes vital because it acts as a foundation for design rationale according to Methodological Rigor in DSR [43].

3.3 Data Collection and Analysis Methods

The process for data collection was conducted in accordance with the system described in the DSR evaluation methodology framework, relevant to both quantitative and qualitative criteria.

3.3.1 Phase 1: Problem Identification (Qualitative)

Semi-structured interviews (N = 18) were administered to software developers, quality assurance engineers, and technical team managers. The problems identified in the developer group include estimation fatigue, intricate legacy systems, and differing interpretations of requirements.

3.3.2 Phase 2: Artifact Validation (Quantitative & Qualitative Matrices)

Evaluations were conducted based on the data collected from each sprint in Jira, assuming that the development process in the traditional agile development paradigm without AI assistance is the base point.

- **Development KPIs:** Coding velocity, defect density, and refactoring quality.
- **QA KPIs:** Test coverage, rate of edge cases discovered, and hallucination rate.
- **Code Review KPIs:** Shortened code review time, manual interaction rate, and vulnerability detection rate.

Post-sprint survey was conducted to evaluate productivity improvement and verification fatigue.

4 Proposed Work

4.1 Problem Validation

The initial design requirements were established based on the Relevance Cycle of the Design Science Research approach. Semi-structured interviews (N = 18) were held with representatives of the industry engaged in the migration of legacy systems.

From the findings above, it became evident that the manual estimation approach is mostly affected by a lack of sufficient information regarding legacy systems leading to estimation fatigue and inconsistencies in planning. Additionally, respondents mentioned the issue related to insufficient transparency of AI-supported workflows and, therefore, low trust in automated suggestions.

4.2 Framework Architecture

The suggested framework involves a multilevel architecture that will distribute tasks among various stages of the Agile lifecycle. This division aims to enhance modularity, avoid cognitive overload, and promote systematic integration of generative AI applications.

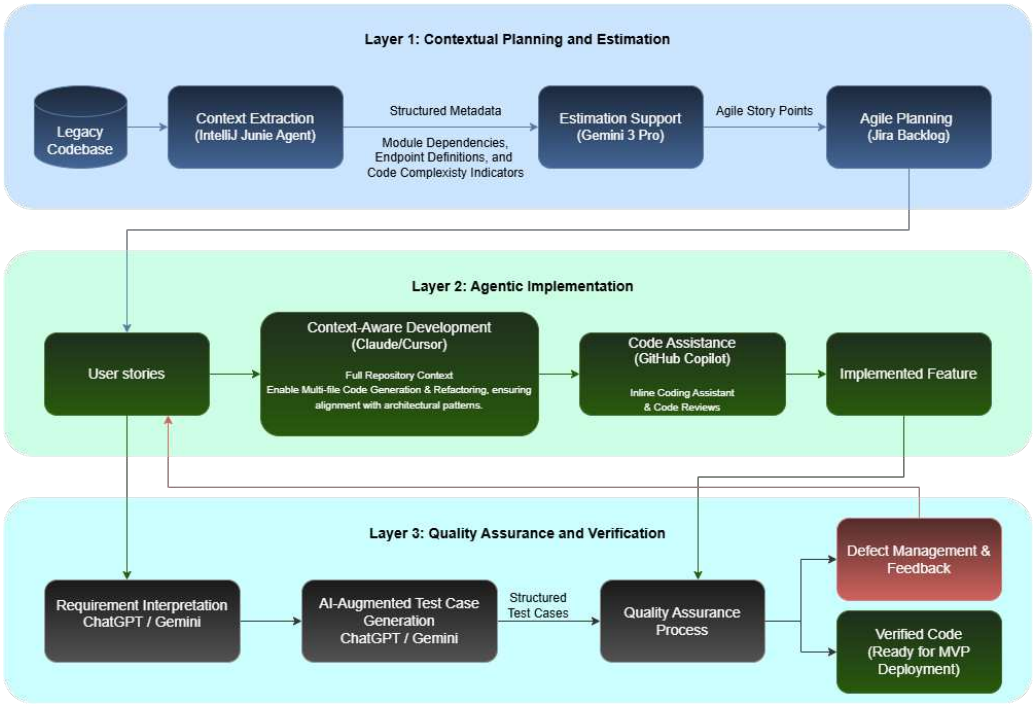


Figure 1: Architecture of the AI-Augmented Agile MVP Framework

The framework consists of three primary layers:

4.2.1 Layer 1: Contextual Planning and Estimation

This layer aids in the Agile Inception stage with requirement comprehension and estimation efforts.

- **Context Extraction:** Agents based on IDEs like JetBrains' Junie can be leveraged to extract structured metadata from legacy code repositories, which include module dependencies, endpoints, and metrics related to code complexity.
- **Estimation Support:** The extracted context information is then leveraged by large language models such as Gemini 3 Pro to provide effort estimations in the form of Agile story points.

4.2.2 Layer 2: Agentic Implementation

It helps during the development process with code generation and refactoring for engineers.

- **Context-Aware Development:** The use of tools like Cursor and Claude is helpful to generate and refactor code across multiple files using the context from the entire repository. This will make sure that there is consistency with the existing architecture patterns.
- **Code Assistance:** GitHub Copilot is a coding assistant tool that helps in generating boilerplate code like DTO, services, unit tests, etc.

4.2.3 Layer 3: Quality Assurance and Verification

It helps in the validation of the code generated and implemented for ensuring its accuracy and validity.

- **Requirement Interpretation:** The AI-based conversational interfaces like ChatGPT and Gemini are used in interpreting the user stories and extracting the structured acceptance criteria from them.

- **Test Case Generation:** AI helps in generating test cases including edge cases, as well as integration testing, in assisting the QA engineering processes.
- **Preliminary Code Review:** It helps in performing an initial review of the code, by identifying basic problems in the code such as format problems, or security vulnerabilities.

5 Results and Discussion

5.1 Phase I: Inception Results

This experiment compared AI-driven estimation methodology with manual estimation by the experienced technical lead. The findings are shown in Table 1.

Table 1: Estimation Accuracy: Hybrid AI vs. Human Expert

Metric	Hybrid AI Workflow	Human Expert	Impact
Avg. Estimate per API	7.5 Hours	8.0 Hours (Buffered)	94% Accuracy
Total Time Taken	2.5 Days	5 Days	50% Faster
Method	Context (IDE) + Logic (LLM)	Manual Tracing	Automation

Discussion: The process of AI-driven estimation demonstrated similar accuracy to human-based assessment, having 94% precision. On the other hand, planning efforts were cut in half, which shows that dividing the process into context extraction (agents in IDE) and reasoning (LLMs) enhances the efficiency of estimation. For lean startups, such division leads to significant savings in planning resources without sacrificing precision.

5.2 Phase II: Agentic Implementation Results

During implementation, different AI tools were assigned to development tasks based on complexity and context requirements. The results are summarized in Table 2.

Table 2: MVP Coding Evaluation (Implementation Phase)

Task Type	AI Tool Used	Story Pts	Actual Hrs	Iterations	Refactor Acc.	Context Overflows
Price Upload APIs	Cursor IDE	5	3.5	1	98%	1
Multi-Tenancy Support	Cursor IDE	8	6.5	2	95%	0
Security Based APIs	Cursor IDE	8	5.0	1	99%	2
Security Master APIs	Cursor IDE	5	3.0	1	98%	0
Base API Structure	GitHub Copilot	3	2.5	4	85%	0
CI/CD Pipeline Setup	GitHub Copilot	5	6.0	6	80%	0
Security Master List API	GitHub Copilot	3	3.5	4	88%	3
Security & Compliance	Claude AI	5	4.5	1	95%	1
Account Upload APIs	Claude AI	5	5.5	2	98%	2

Discussion: From the findings, it is evident that there is a notable gap in performance between deep-context AI agents and conventional IDE agents. The two deep context agents, namely Cursor and Claude, showed remarkable accuracy in terms of refactoring across all cases, achieving an accuracy level ranging between 95% and 99%. On the other hand, GitHub Copilot was adequate in handling boilerplate and structural cases, but not business logic-based refactoring.

5.3 Phase II: AI-Enhanced Quality Assurance Outcomes

Conversational AI tools were used to support requirement interpretation and test generation activities. The results are presented in Table 3.

Discussion: As per the results, it is found that conversational AI has considerably improved the efficiency of QA due to increased speed in the requirement analysis phase and discovery of more edge cases. QA engineers were capable of identifying as many as five edge cases per story, allowing them to not completely depend upon developers for the domain knowledge. Nevertheless, there are cases of hallucination that still need validation by humans.

Table 3: QA Test Generation and Requirement Decoding

Target Test Phase	AI Tool Used	Drafting Speed	Prompts	Edge Cases Found	Hallucinations
Grooming & AC Extraction	ChatGPT	Fast	2	5	0
Functional Integration	Gemini 3 Pro	Fast	3	4	1
Non-Functional / Sanity	ChatGPT	Fast	2	3	2
End-to-End (E2E)	ChatGPT & Gemini	Moderate	4	2	0
Functional API Tests	Gemini 3 Pro	Fast	3	2	0
Security Equivalences	ChatGPT & Gemini	Slow	5	5	1

5.4 Phase II: Automated Code Review & Verification Fatigue

GitHub Copilot was used for automated pull request reviews to assess its effectiveness in vulnerability detection and review acceleration.

Table 4: Automated Code Review Evaluation (Copilot Exclusive)

PR Task Focus	Duration (Mins)	False Positives	Vulns Caught	Arch. Drift Flagged	Override Rate
Security Equivalence	25	14	2	0	45%
Global JPA Auditing	30	12	1	0	40%
Rebalance Template Logic	20	11	0	0	50%
Security Master List	25	10	1	0	45%
Upload Account JSON	35	15	1	0	55%
Rebalance Follow-up Fixes	15	9	0	0	40%

Discussion: While automated tools were able to identify localized threats accurately, there was no global architectural knowledge because the scope of the assessment was restricted to the code diffs alone without taking the entire system into account. The implications of this shortcoming were that the false positive ratio remained very high, and human overrides reached 55%.

5.5 Global Market Validation and External Comparison

The phenomenon of verification fatigue, alongside the high override rates, noted in the fintech setting is consistent with empirical studies done worldwide. Studies on large language models for vulnerability discovery indicate that such advanced models produce many false positives, including instances where they mislabel corrected code. The results reiterate that AI is currently much better suited as a supporting tool in the development process rather than an independent reviewer [45].

In addition, research on generative AI in software engineering indicates that although AI significantly cuts down the efforts needed to do basic coding, it moves cognitive burdens associated with architectural considerations and validation to humans. Such findings further affirm that verification fatigue is a universal challenge in software engineering supported by AI rather than a localized issue within the fintech sector [41].

5.6 Thematic Analysis: Developer Flow and Tool Adoption

Unstructured feedback collected after sprint (N = 12) was analyzed to determine developer productivity and attitudes.

- **Deep-Context vs. Friction:** Participants (P06, P08) using Cursor found the tool to be exceptionally powerful in producing full APIs without any human effort at all. Yet, some (P02) pointed out the opposition involved in deviating from the legacy IDE paradigm.
- **The Verification Shift:** The aforementioned analysis shows that the deployment of AI systems effectively solves the problem of "blank page syndrome." Still, there is no shortage of cognitive effort; the developers (P01, P05) admitted that the verification process requires significant mental efforts.
- **The QA Multiplier:** The Quality Assurance experts (P10, P11, P12) unanimously agreed on the critical importance of conversational language models, pointing out a 40-50% productivity boost.

6 Conclusion and Future Work

6.1 Conclusion

The findings resulting from this Design Science Research clearly show that the inclusion of generative AI in the Lean Startup Build-Measure-Learn process can be highly beneficial provided that it takes place in a structured, agent-driven workflow process, not merely by itself in an ad-hoc manner through tooling.

During the Planning and Estimation stage, the segmentation of context extraction from reason-based estimation resulted in a 50% decrease in planning time, but the estimations were still close to 94% accurate as opposed to estimates made by domain experts. This means that the use of generative AI in the decomposition of Agile tasks positively affects estimation processes.

The analysis during the Implementation stage showed that there is a difference in performance depending on the task complexity and the depth of the required context when evaluating different tools. The systems relying on deep-context such as Cursor performed at up to 98% accuracy levels while dealing with legacy migrations, but inline systems like GitHub Copilot performed better in boilerplate generation tasks.

Conversational AI solutions like ChatGPT and Gemini showcased their usefulness within quality assurance processes. Requirement analysis became more precise, and it became quicker to produce test cases, even detecting new types of edge cases. Thus, dependence on developers' domain expertise decreased, making the process of quality assurance less dependent on human effort.

Nevertheless, one significant issue within the workflow of AI-assisted development was highlighted – verification fatigue. Tools that automatically analyzed the source code were prone to generating false alarms and needed extensive human involvement in overriding them, with the percentage ranging up to 55%. This shows that modern large language models are unable to assess the consistency of the project architecture because they have insufficient access to a global codebase. In other words, human effort is transferred from code writing to its verification and architecture governance but not substituted.

In general, generative AI tools considerably accelerate MVP development and reduce human effort in the early stages of the engineering process; however, continued efficiency requires close supervision by humans in verification and architecture governance.

6.2 Dissertation Contributions

This research seeks to address the gap between what is possible with generative AI and how this can be done with generative AI in high-growth startups. The main findings include:

- **AI-Augmented Agile MVP Framework:** This research offers an end-to-end Design Science Research framework based on the incorporation of multi-agent AI technologies throughout the creation and development process.
- **Empirical Evaluation of AI Tooling:** An empirical analysis of various AI tools, such as Cursor, Claude, GitHub Copilot, ChatGPT, and Gemini in specific Agile applications in a fintech migration case study.
- **AI Tool Performance and QA Integration:** In this research, the performance of various code assistant AI technologies (Cursor, Claude, GitHub Copilot) will be analyzed to determine which tool best suits the development and integration of ChatGPT and Gemini in quality assurance procedures.
- **Identification of Verification Fatigue:** This research demonstrates substantial changes in the labor expected of developers in using GenAI technology. While it significantly decreases the effort required by the developer in writing code, there is still an important need for human validation to avoid any hallucinations or misunderstandings of the requirements.

6.3 Future Work

Conclusions and Recommendations from the Current Study Provide Numerous Opportunities for Further Research, Including but Not Limited to:

1. **Improved AI Code Review Systems:** Design and deployment of lightweight AI-driven models that can help minimize false positives in automatic pull request reviews. Another research area is comparing various types of AI tools dedicated to automated code review.
2. **Long-Term Technical Debt Analysis:** Researching the possibility of accumulating some form of technical debt when creating MVPs via AI support.

3. **Predictive Agile Intelligence:** Integration of Generative AI in Agile intelligence analytics that can predict velocity changes, sprint problems, bottlenecks, and other issues.

4. **Context Extraction Using More Advanced Solutions:** The current experiment utilized Junie as the only tool for context and metadata extraction in Phase I. It would be interesting to use other tools like Claude and Cursor to extract information related to legacy architectures.

References

[1] B. Kazakevich and K. F. Joiner, "Agile approach to accelerate product development using an mvp framework," *Australian Journal of Multi-Disciplinary Engineering*, vol. 20, no. 1, pp. 115–130, 2023.

[2] A. Rezazadeh, M. Kohns, R. Bohnsack, N. António, and P. Rita, "Generative ai for growth hacking: How startups use generative ai in their growth strategies," *Journal of Business Research*, vol. 192, 2025.

[3] World Economic Forum, "Leveraging generative ai for job augmentation and workforce productivity," World Economic Forum, Insight Report, 2024.

[4] R. Yadav, "Transforming project management with ai: Opportunities and challenges," *Open Journal of Business and Management*, vol. 12, pp. 3794–3805, 2024.

[5] A. Marino and P. Pariso, "Rethinking the ai startups creation business and management model," in *2024 IEEE International Conference on Engineering, Technology and Innovation (ICE/ITMC)*. IEEE, 2024.

[6] E. Szathmári, Z. Varga, A. Molnár, G. Németh, Z. P. Szabó, and O. E. Kiss, "Why do startups fail? a core competency deficit model," *Frontiers in Psychology*, vol. 15, 2024.

[7] F. Pattyn, R. Kazmi, P. Goetz, A. Aldhahri, and Aqsa, "Rectv: Enhancing agile decision-making in startups through value-driven requirements prioritization," *IEEE Access*, vol. 13, pp. 142 546–142 560, 2025.

[8] A. K. Gangula, "A comparative analysis of llm-driven vs. manual legacy code refactoring: A case study in .net core migration," *International Journal of Computer Science and Mobile Computing*, vol. 14, no. 9, pp. 80–91, 2025.

[9] P. Tominc, D. Oreski, and M. Rožman, "Artificial intelligence and agility-based model for successful project implementation and company competitiveness," *Information*, vol. 14, p. 337, 06 2023.

[10] R. Mallidi, "Story point estimate model: Project development using generative ai (genai) tools," *International Journal of Computer Applications*, vol. 186, no. 54, pp. 68–71, 2024.

[11] N. Kucuk and M. Yilmaz, "Impact of minimum viable product on software ecosystem," *Information and Software Technology*, 2025.

[12] P. M. Tang, J. Koopman, and S. McClean, "Using augmentation-based ai tool at work: A daily diary study," *Journal of Management*, 2024.

[13] M. Sekli and M. Portuguez Castro, "Fostering entrepreneurial success from the classroom: Unleashing the potential of generative ai," *Education and Information Technologies*, pp. 1–29, 2025.

[14] R. Mo, D. Wang, W. Zhan, Y. Jiang, Y. Wang, Y. Zhao, Z. Li, and Y. Ma, "Assessing and analyzing the correctness of github copilot's code suggestions," *ACM Transactions on Software Engineering and Methodology*, vol. 34, no. 7, pp. 1–32, August 2025.

[15] J. Umbreen, S. Abbas, M. Z. Mirza, A. Naseem, and Y. Ahmad, "Exploring the determinants of sustainable digital startups in a developing country: A qualitative analysis," *IEEE Access*, vol. 13, pp. 166 569–166 580, 2025.

[16] J. Briesmeister and K. Scheidgen, "The minimum viable product (mvp): Theory and practice," *Journal of Management*, 2024.

[17] A. Contigiani and D. H. Hsu, "Artificial intelligence, lean startup method, and product innovations," *Management Science*, 2025, (Articles in Advance).

[18] S. Raneri, F. Lecron, J. Hermans, and F. Fouss, "Predictions through lean startup? harnessing ai-based predictions under uncertainty," *International Journal of Entrepreneurial Behavior & Research*, vol. 29, no. 4, pp. 886–912, 2022.

[19] G. Santoro, F. Jabeen, T. Kliestik, and S. Bresciani, "Ai-powered growth hacking: Benefits, challenges and pathways," *Management Decision*, 2024.

[20] G. Xiang, M. Peng, F. Tang, and Y. Liu, "Unpacking the impact of entrepreneurial learning on business model innovation in internet startups: Mediating roles of digital capabilities," *Technology in Society*, vol. 77, p. 102578, 2024.

[21] F. Zhang, B. Yang, and L. Zhu, "Digital technology usage, strategic flexibility, and business model innovation in traditional manufacturing firms: The moderating role of the institutional environment," *Technological Forecasting and Social Change*, vol. 194, p. 122726, 2023.

[22] C. Gerlich, K. Brenk, J. L. Antonio, D. K. Kanbach, and S. Kraus, "Wielding a double-edged sword: Unravelling the development of dynamic capabilities through growth hacking," *Journal of Business Research*, vol. 189, p. 115172, 2025.

[23] S. Fosso Wamba, M. M. Queiroz, and L. Trinchera, "The role of artificial intelligence-enabled dynamic capability on environmental performance: The mediation effect of a data-driven culture," *International Journal of Production Economics*, vol. 268, p. 109131, 2024.

[24] X. Zhu and Y. Li, "The use of data-driven insight in ambidextrous digital transformation: How do resource orchestration, organizational strategic decision-making, and organizational agility matter?" *Technological Forecasting and Social Change*, vol. 196, p. 122851, 2023.

[25] C. Giardino, N. Paternoster, and M. Unterkalmsteiner, "The use of generative ai tools in the inception stage of software startups," in *Software Business*. Springer, 2025, pp. 34–49.

[26] M. R. Islam and P. Sandborn, "Multimodal generative ai for story point estimation in software development," in *IEEE Conference on Artificial Intelligence (CAI)*. IEEE, 2025.

[27] E. Klotins, M. Unterkalmsteiner, and T. Gorschek, "A progression model of software engineering goals, challenges, and practices in start-ups," *IEEE Transactions on Software Engineering*, vol. 47, no. 3, pp. 498–521, 2021.

[28] H. Petersen and M. Schmidt, "Agentic ide workflows: Evaluating cursor and github copilot in enterprise java refactoring," *Journal of Systems and Software*, vol. 210, 2025.

[29] M. A. Karabıyık, "Refactorgpt: a chatgpt-based multi-agent framework for automated code refactoring," *PeerJ Computer Science*, vol. 11, p. e3257, 2025.

[30] M. S. Fayed and A. S. Fayed, "Prompt-driven development with claude code: Developing a tui framework for the ring programming language," *Electronics*, vol. 15, no. 4, 2026. [Online]. Available: https://www.mdpi.com/2079-9292/15/4/903

[31] A. Moradi Dakhel, A. Nikanjam, V. Majdinasab, F. Khomh, and M. C. Desmarais, "Github copilot ai pair programmer: Asset or liability?" *Journal of Systems and Software*, vol. 203, p. 111734, 2023.

[32] O. Asare, M. Nagappan, and N. Asokan, "Is github's copilot as bad as humans at introducing vulnerabilities in code?" *Empirical Software Engineering*, vol. 28, no. 6, p. 129, 2023.

[33] M. L. Siddiq, J. C. S. Santos, G. Santos, and W. Aljedaani, "Security of ai-generated code: An empirical study on developer perceptions and vulnerability introduction," *Empirical Software Engineering*, vol. 29, no. 4, p. 92, 2024.

[34] M. Micheli, I. Hupont Torres, B. Delipetrev, and J. Soler-Garrido, "The landscape of data and ai documentation approaches in the european policy context," *Ethics and Information Technology*, vol. 25, no. 4, p. 56, 2023.

[35] M. Schäfer, S. Nadi, A. Eghbali, and F. Tip, "An empirical evaluation of using large language models for automated unit test generation," *IEEE Transactions on Software Engineering*, vol. 50, no. 1, pp. 85–105, 2024.

[36] A. Ziegler, E. Kalliamvakou, X. Li, and Others, "Measuring github copilot's impact on productivity," *Communications of the ACM*, vol. 67, no. 1, pp. 54–63, 2024.

[37] A. Sergeyuk, Y. Golubev, T. Bryksin, and I. Ahmed, "Using ai-based coding assistants in practice: State of affairs, perceptions, and ways forward," *Information and Software Technology*, vol. 178, p. 107610, 2025.

[38] V. V. Jensen, A. R. Bruun, and J. S. Persson, "Managing expectations towards ai tools for software development: a multiple-case study," *Information Systems and e-Business Management*, vol. 23, pp. 869–901, 2025.

[39] S. Zhang, Z. Xing, R. Guo, F. Xu, L. Chen, Z. Zhang, X. Zhang, Z. Feng, and Z. Zhuang, "Empowering agile-based generative software development through human-ai teamwork," *ACM Transactions on Software Engineering and Methodology*, vol. 34, no. 6, pp. 156:1–156:46, 2025.

[40] K. Li, F. Zhang, L. Hughes, and M. A. Griffin, "Leveraging generative ai for project management: The role of mindfulness and job crafting," *International Journal of Project Management*, vol. 44, p. 102816, 2026.

[41] M. Salomon, E. Koshchenko, A. Sergeyuk, R. Holmes, G. C. Murphy, and T. Fritz, "From disruptions to discussions: How genai impacts human interactions in software development," *IEEE Transactions on Software Engineering*, 2026.

[42] E. Engström, M.-A. Storey, P. Runeson, and M. Höst, "Constructive master's thesis work in industry: Guidelines for applying design science research," in *Proceedings of the IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*. IEEE, 2021, pp. 1–12.

[43] R. E. Dyers, H. Mahomed, and D. Van Greunen, "Design science research in quality improvement: Embedding rigour in digital health innovation," *African Journal of Primary Health Care & Family Medicine*, vol. 17, no. 2, p. a5194, 2025. [Online]. Available: https://doi.org/10.4102/phcfm.v17i2.5194

[44] N. S. M. Petrin, J. C. Néto, and H. C. Mariano, "Maistro: Towards an agile methodology for ai system development projects," *Applied Sciences*, vol. 15, no. 5, 2025.

[45] G. Antal, L. Értekes, N. Szolnoki, and P. Hegedűs, "Evaluating retrieval-augmented generation for llm-based vulnerability detection: An empirical study on real-world java vulnerabilities," *IEEE Access*, vol. 14, pp. 50 878–50 891, 2026.

A Phase I: Pre-Adoption Interview Guide

A.1 Overview

This structured interview guide was prepared to verify the relevance of the research problem within the organization's setting (Relevance Cycle). The aim was to establish existing bottlenecks in the *Planning and Estimation* phase of the legacy migration project (Jersey to Spring Boot).

A.2 Demographics and Context

1. **Role:** What is your current role in the migration project (e.g., Project Manager, Product Owner, Technical Lead)?
2. **Experience:** How many years of experience do you have of working with Agile methodologies?
3. **Legacy Familiarity:** On a scale of 1–5, how familiar are you with the Jersey framework?

A.3 Current Estimation Challenges

1. **Process Efficiency:** Explain the process you currently follow for estimating a migration user story. How long do refinement meetings usually take?
2. **Legacy Complexity:** Why are manual estimates challenging for Jersey to Spring Boot migration stories?
3. **Cognitive Load:** Does the team suffer from estimation fatigue? Explain how.

A.4 AI Readiness and Requirements

1. **Trust Factors:** If the AI estimates a story point, what data do you need to trust it?
2. **Risks:** What risks do you see in using AI for estimation?

B Phase I: Thematic Analysis Data

Participant Demographics

The qualitative study sample comprised 18 subjects in the Karachi software industry:

- 3 Project Managers (PM)
- 4 Technical Leads / Architects
- 5 Senior Developers
- 3 QA Leads/Engineers
- 2 Product Owners (PO)
- 1 Scrum Master

Selected Raw Responses

- **P01 (Tech Lead):** "It is all about hidden dependencies. We assume that it is just a simple CRUD, but then there are three other services that invoke it." *(Theme: Legacy Obscurity)*
- **P13 (PM):** "It is difficult. We are always lagging in our plans." *(Theme: Inefficiency)*
- **P08 (Tech Lead):** "I am already using Copilot for coding. So why not use it in the planning as well? But it should also give me the Chain of Thoughts" *(Theme: Trust Requirements)*

C Phase II: Evaluation Matrices

Below is the data that has been gathered from the Phase II implementations and QA cycles, all presented quantitatively and qualitatively. It should be noted that all data was anonymized in order to protect organization's confidentiality while preserving all reported metrics.

C.1 Post-Sprint Qualitative Sentiment & Usability

Table 5: Participant Qualitative Feedback Matrix

ID	Role	Tool Evaluated	Usability	Fatigue	Trust	Prod. Δ	Unstructured Feedback Summary
P01	Tech Lead	Copilot (PRs)	4	3	3.5	+25%	Identifies security risks but lacks full context; high noise.
P02	Snr SDE	Claude IDE	4.5	3	4	+35%	High reasoning but struggles with legacy integration out-of-the-box.
P03	Snr SDE	Copilot (IDE)	4.5	4	2	+15%	Frictionless, but core reasoning fails on complex engineering tasks.
P04	Snr SDE	Cursor IDE	3	3	3	+15%	Functional, but perceived productivity did not justify the shift.
P05	Snr SDE	Claude AI	3.5	1	5	+50%	Exceptional reasoning; requires strategic use to manage token limits.
P06	SDE-II	Cursor IDE	3.5	1	5	+80%	Flawless adherence to existing standards; massive efficiency gains.
P07	SDE-II	Copilot (IDE)	4	2	3	+45%	Strong for boilerplate automation and reducing manual toil.
P08	SDE-II	Cursor (Flash)	4.5	2	4.5	+60%	Pre-processing repository graphs yielded excellent context retention.
P09	QA Lead	ChatGPT	4.5	2	4	+30%	Reliable analytical assistant for extracting hidden business logic.
P10	Snr SQAE	ChatGPT	4.5	2.5	4.5	+45%	Dual-model strategy highly effective for edge cases.
P11	SQAE	ChatGPT / Gemini	5	2	4.5	+50%	Indispensable "Swiss Army knife" for daily QA operations.
P12	SQAE	ChatGPT / Gemini	5	2	4.5	+50%	Accelerates testing lifecycles by handling exhaustive drafting.

C.2 MVP Coding Evaluation Matrix

Table 6: MVP Coding Evaluation (Cursor vs. Claude vs. Copilot)

Task ID	Task Summary	Assignee	AI Tool Used	Story Pts	Actual Hrs	Iterations	Accuracy	Defects	Overflows
UCM-29	Develop Price Upload APIs	P06	Cursor IDE	5	3.5	1	98%	1	0
UCM-38	Multi-Tenancy Support	P06	Cursor IDE	8	6.5	2	95%	0	1
UCM-68	Security Based APIs	P04	Cursor IDE	8	5.0	1	99%	0	2
UCM-44	Security Master APIs	P08	Cursor IDE	5	3.0	1	98%	0	0
UCM-9	Base API Structure	P03	GitHub Copilot	3	2.5	4	85%	1	0
UCM-118	CI/CD Pipeline	P07	GitHub Copilot	5	6.0	6	80%	0	0
UCM-208	GET Security Master List	P03	GitHub Copilot	3	3.5	4	88%	1	3
UCM-15	Security & Compliance	P05	Claude AI	5	4.5	1	95%	0	1
UCM-16	CICD Pipeline Setup	P02	Claude AI	3	2.0	1	96%	0	0
UCM-115	Account Upload APIs	P05	Claude AI	5	5.5	2	98%	2	2

C.3 QA Test Generation Matrix

Table 7: QA Test Generation (Gemini & ChatGPT)

Task ID	Task Summary	Assignee	AI Tool Used	Target Phase	Prompts	Speed	Edge Cases	Flaky	Hallucinations
N/A	Grooming & AC Extraction	P09	ChatGPT	Req. Analysis	2	Fast	5	0	0
UCM-152	Upload & Save APIs	P10	Gemini 3 Pro	Functional	3	Fast	4	0	1
UCM-179	Multi-tenancy Verif.	P10	ChatGPT	Sanity	2	Fast	3	0	2
UCM-71	Import Account APIs	P11	Both	E2E	4	Mod	2	1	0
UCM-89	Create Security APIs	P11	Gemini 3 Pro	Functional	3	Fast	2	0	0
UCM-315	Security Equivalence POST	P12	Both	Integration	5	Slow	5	2	1

C.4 Automated Code Review Evaluation

Table 8: Automated Code Review Evaluation (GitHub Copilot Exclusive)

PR ID	PR Summary	Reviewer	Duration (Mins)	False Positives	Vulns Caught	Drift Flagged	Override Rate
PR 77	Create Security Equivalence	P01	25	14	2	0	45%
PR 84	Implement Global JPA Auditing	P05	30	12	1	0	40%
PR 75	Account API (Template Logic)	P03	20	11	0	0	50%
PR 74	Migrate Security Master List	P06	25	10	1	0	45%
PR 69	Account API (Upload JSON)	P02	35	15	1	0	55%
PR 75	Account API (Follow-up Fixes)	P07	15	9	0	0	40%

C.5 Thematic Analysis: Selected Raw Responses

In order to add further context to the matrices mentioned above, below are the following unstructured responses that have been extracted from Phase II implementations in order to illustrate the main concerns:

- **P01 (Tech Lead - Copilot PRs):** "Shows great potential for uncovering hidden threats and verifying PR descriptions... The context window, however, can only span the files contained within the PR, which results in mild hallucinations... The tool greatly cuts down on early code reviews but is certainly not meant to replace human-driven architectural reviews." *(Theme: The Verification Shift & Context Limitations)*
- **P05 (Senior SDE - Claude AI):** "Exceptional reasoning and near 100% accuracy on 'Max' effort settings eliminate the need for intense code verification... However, severe token consumption rapidly exhausts standard limits, halting workflow... Maximum productivity requires a pay-as-you-go token model and strategic use." *(Theme: Advanced Reasoning vs. Resource Bottlenecks)*
- **P06 (SDE-II - Cursor IDE):** "Its deep repository context allows for flawless adherence to existing codebase standards, enabling the automated creation of full APIs... with zero manual intervention. However, overall usability is hampered by the necessity to abandon the IntelliJ ecosystem for backend profiling..." *(Theme: Deep-Context vs. Ecosystem Friction)*
- **P08 (SDE-II - Cursor IDE):** "Bypassed standard high-cost bottlenecks by routing the IDE through a lightweight model (Gemini 3 Flash) and proactively leveraging the IDE's codebase indexing... This pre-processing resulted in excellent context retention, drastically reduced token consumption, and minimized hallucinations." *(Theme: Proactive Configuration & Cost Management)*
- **P11 (SQAE - ChatGPT/Gemini):** "An indispensable 'Swiss Army knife' for daily QA operations... deeply integrated into their daily workflow for generating test scenarios, drafting bug reports, and standardizing documentation. The reliance on the tool as a universal utility indicates exceptional usability." *(Theme: The QA Multiplier)*