

**Bachelor's Programme in Industrial Engineering and Management**

# Artificial Intelligence and Minimum Viable Product

Utilizing AI Techniques Across Different Phases of MVP Development in Software Engineering

---

**Taulant Smakiqi**

**Bachelor's Thesis  
2025**

Copyright ©2025 Taulant Smakiqi

**Author** Taulant Smakiqi

---

**Title of thesis** Artificial Intelligence and Minimum Viable Product

---

**Programme** Bachelor's Programme in Science and Technology

---

**Major** Industrial Engineering and Management

---

**Thesis supervisor** Prof. Risto Rajala

---

**Thesis advisor** Fabiana Mendes, PhD

---

**Date** 14.12.2025

**Number of pages** 32 + 4

**Language** English

---

### **Abstract**

Artificial Intelligence (AI) is increasingly integrated into software engineering, yet its role in supporting Minimum Viable Product (MVP) development remains underexplored. MVP development emphasizes rapid iteration, minimal resource use and early user feedback. Understanding how AI techniques contribute to these activities can clarify their value in early-stage software projects.

This thesis investigates how AI techniques support the phases of MVP development, how trustworthy and accurate the techniques are and finally examines their effect on development efficiency.

A literature review was conducted using forward snowballing from an established review on AI in software engineering. 23 peer-reviewed studies were selected for the review. The analysis focused on mapping AI techniques to MVP development phases and assessing the reported accuracy and efficiency outcomes.

The result show that specific AI techniques align with distinct MVP activities. The techniques demonstrate high accuracy results compared to established work models. Efficiency gains arise from automating repetitive tasks, accelerating documentation and modeling activities, optimizing resources and reducing development cycle time. However, most studies were conducted in a laboratory setting, lacking real-world validation.

AI techniques can enhance MVP development by supporting core activities with high accuracy and improving efficiency. Nonetheless, the trust in these tools depend on data quality, model transparency and real-world validation. The limited industrial studies and the absence of research on Large Language Models and Generative AI highlights opportunities for future work.

---

**Keywords** software engineering, minimum viable product, artificial intelligence, natural language processing, machine learning, deep learning, optimization algorithms, expert systems

---

**Tekijä** Taulant Smakiqi

---

**Työn nimi** Pienimmän Toimivan Tuotteen Optimointi Tekoälyllä

---

**Koulutusohjelma** Teknistieteellinen kandidaattiohjelma

---

**Pääaine** Tuotantotalous

---

**Vastuopettaja** Prof. Risto Rajala

---

**Työn ohjaaja** Fabiana Mendes, TkT

---

**Päivämäärä** 14.12.2025      **Sivumäärä** 32 + 4      **Kieli** Englanti

---

### Tiivistelmä

Tekoälyä integroidaan jatkuvasti ohjelmistokehitykseen. Sen roolia pienimmän toimivan tuotteen (engl. Minimum Viable Product, MVP) kehityksessä ei kuitenkaan ole tutkittu. MVP:n kehityksessä korostuu nopea iterointi, pienet resurssit sekä aikainen käyttäjäpaalaute tuotteesta. Erilaiset tekoälytekniikat voivat tehostaa näitä ohjelmistoprojektien varhaisia kehitysvaiheita.

Tämän kandidaatintyön tavoite on tutkia, miten tekoälytekniikat tukevat MVP:n kehityksen eri vaiheita, selvittää tekniikoiden tarkkuus sekä luotettavuus ja tutkia sen vaikutusta kehityksen tehokkuuteen.

Tutkimus toteutettiin kirjallisuuskatsauksena. Aiemman tekoälyn ja ohjelmistokehityksen integraatiota käsittelevän katsauksen pohjalta toteutettiin eteenpäin suuntautuva lumipallohaku, jonka avulla valikoitui 23 tutkimuspapereita. Aineistossa analysointiin, miten tekoälytekniikat sijoittuvat eri MVP:n kehityksen vaiheisiin sekä kuinka luotettavia ja tehokkaita tekniikat ovat.

Katsaus osoittaa, että tietyt tekoälytekniikat sijoittuvat luontaisesti eri MVP:n kehityksen vaiheisiin. Useat tekniikat osoittavat korkeita tarkkuusarvoja, ja tekoäly tehosti useita rutiinitehtäviä ja paransi resurssien käyttöä. Tulosten yleistettävyyttä rajoittaa se, että suurin osa tutkimuksista suoritettiin kontrolloiduissa olosuhteissa.

Tekoälytekniikat voivat tuoda merkittäviä hyötyjä MVP:n kehitykseen parantamalla sekä työn tarkkuutta että tehokkuutta. Tekniikat keventävät manuaalista työtä, nopeuttavat MVP:n iterointia ja auttavat tekemään perusteltuja päätöksiä ohjelmistoprojektien varhaisissa vaiheissa. Tekniikoiden luotettavuus riippuu kuitenkin datan laadusta ja mallien läpinäkyvyydestä. Soveltavan tutkimuksen rajallisuus sekä suurten kielimallien ja generatiivisen tekoälyn puute korostavat tarvetta jatkotutkimukselle.

---

**Avainsanat** ohjelmistokehitys, pienin toimiva tuote, tekoäly, luonnollisen kielen käsittely, koneoppiminen, syväoppiminen, optimointi algoritmit, asiantuntijajärjestelmät

---

## Table of Contents

<b>Preface</b> .....	<b>7</b>
<b>Symbols and abbreviations</b> .....	<b>8</b>
<b>Abbreviations</b> .....	<b>8</b>
<b>1 Introduction</b> .....	<b>9</b>
<b>2 Literature review</b> .....	<b>10</b>
<b>2.1 MVP Development</b> .....	<b>10</b>
2.1.1 Planning .....	10
2.1.2 Requirements .....	11
2.1.3 Design .....	11
2.1.4 Development .....	11
2.1.5 Validation .....	12
<b>2.2 Artificial Intelligence</b> .....	<b>12</b>
2.2.1 Modern AI and its Rapid Development .....	13
2.2.2 Use of AI .....	13
2.2.3 AI in Society .....	14
2.2.4 Ethics .....	14
2.2.5 AI techniques in Software Engineering Context .....	15
<b>3 Methodology</b> .....	<b>17</b>
<b>3.1 Search Strategy</b> .....	<b>17</b>
<b>3.2 Selection Strategy</b> .....	<b>17</b>
<b>3.3 PRISMA framework</b> .....	<b>18</b>
<b>3.4 Supporting Tools</b> .....	<b>19</b>
<b>4 Results</b> .....	<b>20</b>
<b>4.1 Overview of Selected Studies</b> .....	<b>20</b>
<b>4.2 AI techniques that Support MVP development Activities (RQ1)</b> .....	<b>21</b>
4.2.1 AI Support in the Planning phase .....	22
4.2.2 AI Support in the Requirement phase .....	23
4.2.3 AI Support in the Design phase .....	24
4.2.4 AI Support in the Development phase .....	24
<b>4.3 Trust and Accuracy of AI techniques (RQ2)</b> .....	<b>25</b>
4.3.1 Accuracy Metrics and Performance .....	25
4.3.2 Trust and Limitations of AI techniques .....	26
<b>4.4 Efficiency Gains from AI Techniques (RQ3)</b> .....	<b>27</b>
4.4.1 Automation of Repetitive Tasks .....	27
4.4.2 Reducing Time in Development Cycles .....	28
4.4.3 Resource Optimization and Developer Productivity .....	28
<b>5 Discussion</b> .....	<b>30</b>
<b>5.1 Supporting Iterative Development in MVP</b> .....	<b>30</b>
<b>5.2 Real-World Validation</b> .....	<b>30</b>
<b>5.3 Limited Coverage of Large Language Models and Generative AI</b> .....	<b>30</b>
<b>6 Conclusion</b> .....	<b>31</b>

**6.1 Key Findings .....31**  
**6.2 Limitations.....31**  
**6.3 Future Research .....32**  
**References .....33**

## **Preface**

I want to thank my advisor Fabiana Mendes and my supervisor Risto Rajala for their continuous support and guidance.

Otaniemi, 14 December 2025  
Taulant Smakiqi

# Symbols and abbreviations

## Abbreviations

MVP	Minimum Viable Product
SE	Software Engineering
AI	Artificial Intelligence
ML	Machine Learning
DL	Deep Learning
NLP	Natural Language Processing
ES	Expert Systems
OA	Optimization Algorithms
LLM	Large Language Model

# 1 Introduction

Software Engineering (SE) continues to evolve as new technologies become available, and Artificial Intelligence (AI) is one of the most recent tools helping to improve how software is built. AI includes techniques such as machine learning and Natural Language Processing that automate tasks in SE (Alenezi & Akour, 2025). These techniques can be applied across the SE lifecycle.

At the same time, the concept of the Minimum Viable Product (MVP) has become a common way to approach early product development. An MVP is a basic version of a product that includes only the most important features. It is used to test the product with real users and gather feedback early. The main goal is to learn what users need as quickly as possible while keeping time and costs low, allowing teams to iterate rapidly before committing significant resources to full development (Ries, 2011).

Today, a wide range of AI tools are available to software development teams (Durrani et al., 2024). MVP projects often involve limited time and resources, which makes AI a potentially valuable tool in this context. Gaining insight into how AI can support MVP-related tasks may enhance team efficiency and enable more informed decision-making in the early stages of development.

There is already extensive research on how AI techniques can support different software engineering tasks, such as writing requirements, designing systems, developing software, automating tests, and maintaining the software (Durrani et al., 2024). Other studies have focused on how MVP are used in software development and what are the roles of MVPs in this context (Nguyen Duc & Abrahamsson, 2016). However, these two topics, AI in software engineering and MVP development, are studied separately.

This thesis bridges this gap by conducting a literature review guided by the following research questions:

- RQ1: How can AI techniques support these MVP development activities?
- RQ2: To what extent can we trust the accuracy of AI techniques in MVP development?
- RQ3: Do AI techniques increase the efficiency of MVP development?

Therefore, the objective of this thesis is to explore how AI techniques can be utilized across the various phases of MVP development in SE and to assess their accuracy and efficiency benefits.

This thesis is organized into six main chapters. Following the introduction, Chapter 2 presents the literature review, which establishes the concepts needed to understand the intersection of AI and MVP development. Chapter 3 outlines the methodology employed in this study. Chapter 4 presents the results and addresses the three research questions. Chapter 5 discusses the key findings, and Chapter 6 concludes the thesis by summarizing the main results, outlining the limitations, and proposing directions for future research.

## 2 Literature review

This chapter provides a foundation for understanding what a Minimum Viable Product is, how it is used in software engineering, and the main phases involved in MVP development. It then discusses Artificial Intelligence, including its development, common applications, and societal and ethical impacts. Finally, the chapter examines how AI is applied within software engineering.

### 2.1 MVP Development

MVP development is an approach originating from the Lean Startup methodology. This concept was popularized through Eric Ries's lean startup framework (2011). The minimum viable product is not only a stripped-down or simplified version of a final product, but rather an experimental learning tool designed to validate business assumptions with minimal resource investment. Ries emphasizes that the primary objective of an MVP is not to test product design or resolve technical issues alone, but to systematically examine fundamental business hypothesis (Ries, 2011).

At the core of the Lean Startup methodology lies the Build-Measure-Learn feedback loop, in which the MVP plays a crucial role. It represents an iteration of a product that is built quickly and purposely with only essential features required to deliver products core value proposition. Each feature included in the MVP is intentionally selected to test a specific assumption about user behavior or market demand. This is the fundamental characteristics that sets the MVP apart from a prototype or traditional product development. Non-essential features and polish are deliberately left out until proven necessary for user validation or further learning. By excluding non-essential features and refinement, the MVP enables accelerated learning, fast user feedback, and informed decision making in software engineering teams. New features and improvements to the product are introduced only when they have been validated by real-world users and empirical data. In this way, MVPs support the Lean Startup methodology in product development (Ries, 2011).

In the context of software engineering, MVP development adapts the standard development cycle to fit this experimental nature. While Durrani et al. (2024) identify seven classical phases of software engineering; planning, requirement engineering, design, development, testing, deployment and maintenance; an MVP requires a more focused scope. Duc and Abrahamsson (2016) characterize MVPs in startups as early-stage artifacts that are used primarily for prototyping and learning, rather than long-term operation. Aligning with this focus on early validation, this paper defines five essential phases for MVP development: planning, requirements engineering, design, development and validation. This framework excludes the phases of testing, deployment and maintenance presented by Durrani et al. (2024), as these are typically associated with established products.

#### 2.1.1 Planning

In this initial phase, the team pinpoints the core problem to solve and analyzes if there is a real market need for a solution. Planning for a MVP is kept very lightweight, instead of detailed project plans, it often involves quick brainstorming sessions or simple frameworks

(such as Planning Poker) to outline ideas (Alonso et al., 2023). The focus is on formulating key hypotheses about the product: identifying assumptions that need testing and understanding stakeholder expectations. Researchers note that planning in MVP development is informal and fast, aiming to produce just enough of a plan to proceed. The MVP by definition is a barebones version of the product that delivers only the core features needed for early users, and it is built quickly to begin testing those underlying assumptions (Lenarduzzi & Taibi, 2016).

### **2.1.2 Requirements**

In MVP projects, requirements are scoped to only the most essential features that deliver customer or stakeholder value. Typically, the feature list is kept minimal and prioritized by what will validate the product idea. Rather than extensive documentation, teams capture needs as simple user stories or even just assumptions about what users might want. Studies show that in startups, most requirements come from the internal team's own ideas and vision (since at the start there are few real users to ask) (Alonso et al., 2023) (Tripathi, 2019). These requirements are usually recorded in informal formats, for example, as quick notes, spreadsheets, or on a whiteboard, rather than formal requirement specs (Tripathi, 2019). Importantly, requirements in an MVP are evolving, the team expects they will change once actual user feedback comes in. In fact, it is common that an initial set of features defined internally can change completely after the first users or testers react to the MVP (Alonso et al., 2023).

### **2.1.3 Design**

The design phase for an MVP is intentionally minimal and iterative. Instead of comprehensive architecture documents or detailed UML diagrams, the team might create simple mock-ups or wireframes to visualize the user interface and flows. The goal is to design just enough for a working prototype of the idea. Often, startups practice prototyping as their design approach: building a quick model of the product to see how it might look and feel. Nguyen Duc et al. observe that MVP-oriented startups use both throwaway prototypes and evolutionary prototypes. A throwaway prototype is a temporary design, built to learn from and then discarded, whereas an evolutionary prototype is structured so that it can gradually be extended into the actual product. (Nguyen Duc & Abrahamsson, 2016) In both cases, the design is done with the expectation of change. The team continuously refines the design through short feedback loops, knowing the first design is not final (Nguyen Duc & Abrahamsson, 2016). This lean design approach ensures that the architecture remains as simple as possible, enough to support the MVP's limited scope without over-engineering. Simple sketches, wireframes or paper prototypes are used to create a shared understanding of the MVP, rather than heavy documentation (Alonso et al., 2023).

### **2.1.4 Development**

In developing the MVP, the team implements only the core features that are necessary to test the product's value proposition. The emphasis is on speed and functionality, not on building a fully polished or scalable system (Nguyen Duc & Abrahamsson, 2016). In some MVPs, certain features are not developed at all, instead, the team uses a "Wizard of Oz" approach, the product's interface looks like a real, automated system, but behind the scenes a human is manually performing the tasks (Nguyen Duc & Abrahamsson, 2016). This technique simulates the software's behavior without full implementation. Overall, the MVP build

is barebones and not concerned with robustness or optimization. As Lortie et al. (2024) describe, an MVP is essentially the simplest functional version of the product, focusing on delivering the primary user value and nothing more. The team prioritizes validation over perfection. In other words, minor compromises in quality are acceptable at this stage to quickly test assumptions and learn.

### **2.1.5 Validation**

Validation for an MVP is very different from quality assurance in a traditional software project. MVP validation is all about user feedback and rapid learning. The team conducts basic sanity and usability tests to ensure the core features work well enough for users to try, but more importantly they release the MVP to a small group of real users and observe their behavior. Techniques like A/B testing might be used (presenting users with slightly different versions to see which performs better) to answer specific questions about user preferences, and other metrics are collected to validate or reject the team's hypotheses (Alonso et al., 2023). What matters is what the team can learn from how people use the product, not achieving zero-bug software at this stage (Lenarduzzi & Taibi, 2016). Thus, every issue found by a user or any confusion observed is treated as valuable guidance for iteration, rather than something to be thoroughly polished via traditional quality assurance processes. The validation phase for an MVP is considered successful if it yields actionable insights (Alonso et al., 2023). However, this phase is excluded from the scope of this review since validation relies primarily on qualitative user feedback and business hypothesis testing rather than a structured engineering task.

## **2.2 Artificial Intelligence**

Artificial Intelligence refers to machines that operate autonomously and can compute how to act effectively in a wide variety of situations (Russel & Norvig, 2021, p.19). In the last decade, breakthroughs in computing power, data and algorithms have fueled AI progress. Recent advances in AI methods have led to a major increase in the use of AI across the software industry (Martínez-Fernández et al., 2022).

According to Russel and Norvig (2021, p. 19), AI is “not just understanding but also building intelligent machines that can compute how to act effectively and safely in a wide variety of novel situations”. AI has historically been approached from four distinct perspectives: acting humanly, thinking humanly, thinking rationally and acting rationally (Russel & Norvig, 2021, p. 20).

For the first perspective, acting humanly, the Turing test approach was applied to detect if machines performing tasks indistinguishably from humans. In relation to thinking humanly, the researchers usually apply the cognitive modelling approach which aims to replicate human thinking. The thinking rationally approach emphasizes logical reasoning and inference. Finally, in the acting rationally approach, the rational agent approach focusing on agents that act to achieve best outcomes (Russel & Norvig, 2021, p. 22).

The rational agent approach has become the leading paradigm in modern AI, where “AI has focused on the study and construction of agents that do the right thing”. This perspective

emphasizes building systems that can make optimal decision given their knowledge and computational constraints (Russel & Norvig, 2021, p. 22).

### **2.2.1 Modern AI and its Rapid Development**

AI has gone through remarkable transformation since its rise, evolving through distinct phases that have shaped today's landscape. The field began with the work of McCulloch and Pitts in 1943, who proposed artificial neural networks, and Alan Turing's 1950 paper introducing the now infamous Turing test. In 1956 the Dartmouth Conference formally established AI as a field, with John McCarthy coining the term "artificial intelligence" (Russel & Norvig, 2021, p. 35).

The modern era has been marked by several key developments. The integration of probabilistic reasoning and machine learning, the emergence of big data approaches, and most recently, deep learning that has transformed computer vision, natural language processing and other AI domains (Russel & Norvig, 2021, p. 44).

Today's AI is characterized by unprecedented growth and impact. As Russell and Norvig (2021, p. 19) note, "Surveys regularly rank AI as one of the most interesting and fastest-growing fields, and it is already generation over a trillion dollars a year in revenue". AI expert Kai-Fu Lee predicts that its impact will be "more than anything in the history of mankind".

The rapid development is driven by several factors. First, the availability of specialized hardware like GPUs, TPUs and quantum computing has dramatically risen the computing power which in turn has enabled training of increasingly complex models. From 2012 to 2018 there was a 300 000-fold increase in computing power used for machine learning (Russel & Norvig, 2021, p. 33).

The explosive availability of data, and as an extension big data, has provided the fuel for modern AI systems. Russell and Norvig (2021, p. 44) emphasize that these data sets include "trillions of words of text, billions of images, and billions of hours of speech and video".

Algorithmic advanced and breakthroughs techniques in deep learning, and particularly in neural networks and transformer architectures, have revolutionized pattern recognition and language understanding (Russel and Norvig, 2021, p. 44)

### **2.2.2 Use of AI**

In the modern times AI is increasingly embedded in everyday technologies. Image search and recognition powered services allow users to search for images by uploading a photo (reverse image search), auto-tagging people in personal photo albums or even identifying products, plants or artwork from photos. These tools rely on deep learning models for image recognition, and in some contexts, such as medical diagnosis, AI systems have surpassed human-level performance (Russel & Norvig, 2021, p. 46).

Tools such as Apple's Siri or Alphabet's Google Assistant use AI for speech recognition. Natural language understanding and automated responses, helping users in everyday tasks such as setting reminders or searching for information (Russel & Norvig, 2021, p. 47).

With the popular increase in AI chatbots text generation has become a regular tool to complete emails, summarize texts and specially in online customer support. These chatbots use natural language processing to understand questions and generate human-like responses (Russel & Norvig, 2021, p. 20).

AI is also used for recommendation systems, in platforms such as streaming systems (Netflix, YouTube). The platform uses AI to analyze user preferences and suggest new content. AI-based translation services such as Google Translate break down language barriers in real time allowing for people to connect across cultures (Russell & Norvig, 2021, p. 47).

### **2.2.3 AI in Society**

AI is increasingly transforming different aspects of modern society, with both clear benefits and risks. As highlighted by Paić & Serkin (2025), AI has the potential to drive progress in areas such as education, healthcare and scientific progress. However, the same authors warn that labor displacement is a substantial risk: recent analysis have estimated that “around 40% of all working hours could be impacted by AI LLMs such as ChatGPT-4”. Paić & Serkin argue that if AI integration is not properly managed, the AI revolution could deepen inequality and destabilize established industries, leaving workers without meaningful employment.

In healthcare sectors, the World Health Organization advocates caution to protect safety and warns that data and model bias can deepened racial, gender and geographical differences, specifically when representative datasets and trustworthy data is short. Such biases may influence healthcare access and treatment outcomes. In the educational sector, AI demonstrates significant potential for positive transformation through personalized learning systems. Research by Thimmanna (2024) reveals that AI-driven personalized learning paths can considerably improve academic performance, engagement and retention. These systems adapt to individuals learning styles, paces and preferences. The study found that AI-driven curricula demonstrated higher levels of understanding and retention compared to traditional methods. However, Paić & Serkin emphasize that an over-reliance on automated systems and tools risk corroding human cognitive skills overtime. As automation handles more tasks, the capacity for independent thought, critical reasoning and problem-solving skills may diminish. This applies to academic and professional contexts alike.

Beyond sectoral impacts, AI also threatens information integrity and democratic processes. Schlicht (2025) finds that some generative models, including GPT-4o, are more prone to harmful political disinformation, especially during election cycles, with adversarial prompts that mask intent increasing the odds of dissemination.

At the global scale, Paić & Serkin discuss the risk of widening the “digital divide”, gap between populations with ready access to advanced technologies and without. Even highly developed countries are under strain with AI’s computing power and water demands required by large scale systems, putting further pressure on infrastructure, inclusivity and environment. These risks link directly to AI’s footprint, Google’s operational carbon emission increased around 50% from 2019 to 2023, and water demand increased around 20%, both attributed to AI workloads (Sidorkin, 2025).

### **2.2.4 Ethics**

The ethics of artificial intelligence is a complex and increasingly important issue as AI systems become more common in everyday life, industry, and society. It is essential that AI is designed and used in ways that create positive outcomes and reduce potential harm. AI has many beneficial uses, such as improving healthcare, helping in emergencies, supporting people with disabilities, monitoring the environment, and enhancing global communication. However, AI also poses serious risks. It can increase social inequality, enable large-scale surveillance, reinforce existing biases, and be used in dangerous technologies like autonomous weapons (Russel & Norvig, 2021, p. 1037).

Ethical guidelines for AI include key principles such as safety, fairness, transparency, accountability, privacy, inclusivity and preventing misuse. To support these principles, both organizations and individuals must take responsibility. Since AI is used in many different areas and affects diverse groups, ethical approaches must be practical and tailored to specific situations (Russel & Norvig, 2021, p. 1038).

### **2.2.5 AI techniques in Software Engineering Context**

The integration of AI into software engineering is driven by several compelling factors. First, the increasing complexity of software systems demands automated approaches to manage development, testing, and maintenance activities. As Russell and Norvig (2021, p. 19) observe, "AI currently encompasses a huge variety of subfields" and "is relevant to any intellectual task; it is truly a universal field".

Second, the distinction between "AI for SE" (using AI to improve software engineering processes) and "SE for AI" (applying software engineering principles to build AI systems) has become crucial. This thesis focuses on AI for SE, exploring how artificial intelligence techniques can be leveraged to enhance traditional software engineering activities (Sofian et al., 2022).

This review will study 5 AI techniques that are particularly relevant in the domain of software engineering.

Machine Learning (ML) helps systems learn from data and is widely used for predicting bugs, estimating effort, and analyzing code. Deep Learning (DL), a form of ML, supports tasks like code summarization and understanding natural language requirements. Natural Language Processing (NLP) allows AI to work with human language and is used for analyzing requirements and software documentation. Expert Systems (ES) use rule-based knowledge to support decision-making. Optimization Algorithms (OA) help improve software testing, configuration, and resource use (Durrani et al., 2024).

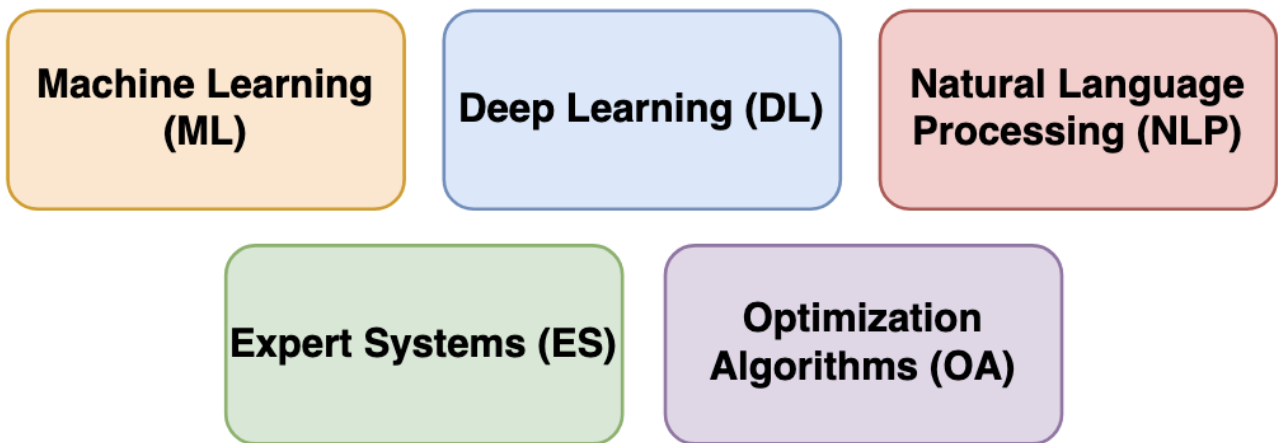


Figure 1 AI techniques studied in this review

This foundation of AI concepts and techniques provides the context for understanding how these tools can be specifically applied to Minimum Viable Product development, which represents a critical phase in modern software engineering where efficiency, speed, and intelligent decision-making are paramount.

### 3 Methodology

As discussed in Chapter 1, this study is a Literature Review (LR) with the goal to investigate how AI can enhance different phases of MVP development. The objective is to map AI tools and techniques that can be used to improve MVP development activities to software engineering lifecycle phases, as well as investigate these tools and techniques accuracy and efficiency.

To reach this goal three research questions (RQs) were developed:

RQ1 - How can AI techniques support MVP development activities?

RQ2 - To what extent can we trust the accuracy of AI techniques in MVP development?

RQ3 - Do AI techniques increase the efficiency of MVP development?

From these research questions, we derived keywords and synonyms that will support the execution of the LR.

<b>Keyword</b>	<b>Synonyms</b>
AI techniques	AI, AI tools, DL, LLM, ML, NLP, artificial intelligence, deep learning, expert system, large language model, machine learning
Accuracy	precision, quality, reliability, validity
Efficiency	optimization, performance improvement, productivity, speed
Software Engineering	programming, software development, software engineering, coding

Table 1 Keywords and Synonyms

The keywords presented on Table 1 helped on the screening of the pre-selected papers. These terms guided the selection of articles by capturing the core concepts of AI techniques and software engineering. Using multiple synonyms for each concept helped to account for differences in terminology across studies. The next section details the search strategy employed in this study.

#### 3.1 Search Strategy

The search for relevant papers was primarily based on snowballing of the papers included in Durrani et al. (2024)'s literature review.

Forward snowballing was conducted on 159 papers over two iterations. In the first iteration, Perplexity, a Large Language Model (LLM) specializing in research was used to screen the papers by applying the inclusion and exclusion criteria presented in table 2. The results were manually reviewed to ensure appropriateness. The second iteration was conducted manually to identify the most relevant papers for inclusion in the review.

#### 3.2 Selection Strategy

Inclusion and exclusion criteria were applied to select studies that address the research questions of this study. The inclusion criteria ensured that only studies focusing on AI applications in software engineering were considered. These criteria were applied to the studies retrieved during the snowballing phase.

	Inclusion Criteria (IC)		Exclusion Criteria (EC)
IC1	Articles about the applications of AI for software engineering	EC1	Articles with four or less pages
IC2	Peer-reviewed articles	EC2	Articles unrelated to SE or MVP
IC3	Articles include keywords and synonyms	EC3	Articles that do not meet the inclusion criteria
IC4	Articles in English	EC4	Articles with unclear results
IC5	Article is available in Aalto Primo	EC5	Article does not relate to the RQ's

Table 2 Inclusion and Exclusion Criteria

### 3.3 PRISMA framework

A PRISMA flow diagram was used to document the screening and selection process, including the number of records identified, screened, assessed for eligibility, and included in the final review.

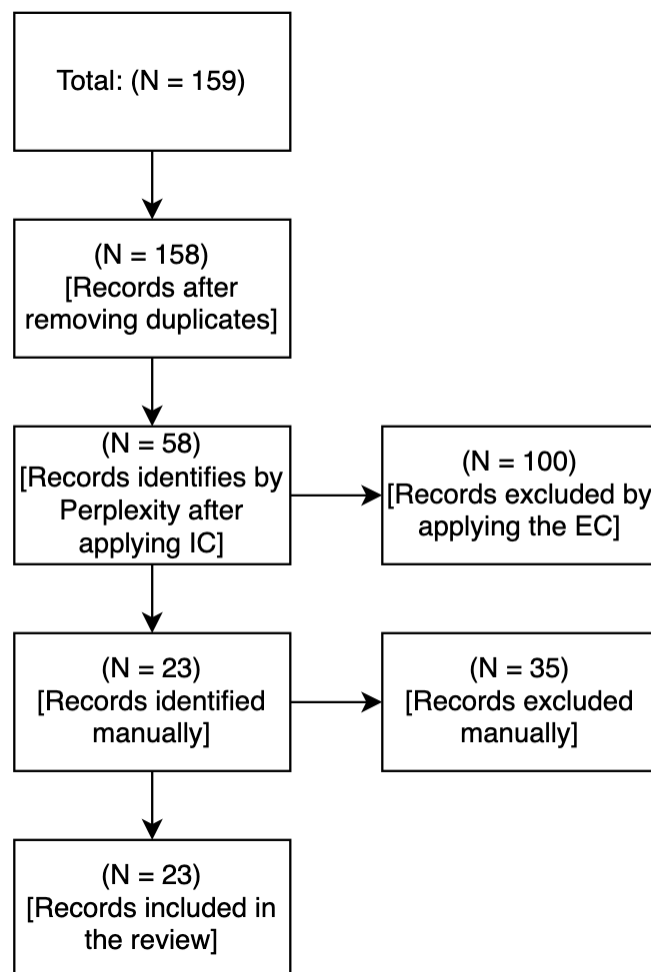


Figure 2 Prisma Framework

### **3.4 Supporting Tools**

Parsifal was used to organize the methodology and manage the review process. Zotero, a reference management tool, helped to store, organize, and cite the identified articles. ChatGPT 5, a LLM, was employed to brainstorm keywords and synonyms to guide the literature search. Perplexity assisted in iterating through the material to identify potentially relevant articles. All articles identified by the LLM were subsequently manually reviewed to confirm their relevance to the study.

## 4 Results

### 4.1 Overview of Selected Studies

A total of 23 papers were selected for the literature review after applying the defined inclusion and exclusion criteria. These papers form the empirical basis for examining how different AI techniques contribute to MVP development across the software engineering lifecycle. Figure 3 presents the temporal distribution of the selected papers, showing that there is a notable increase of publications after 2020.

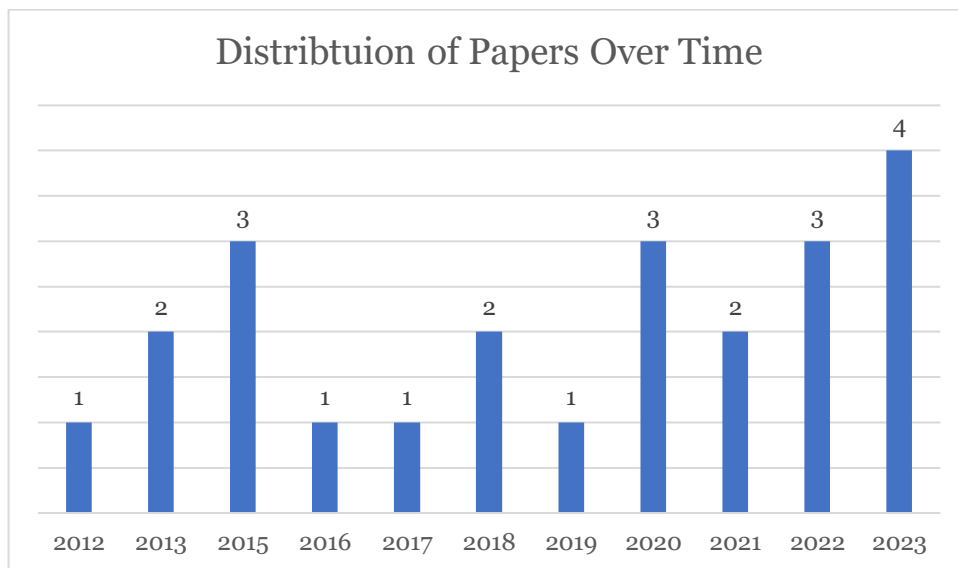


Figure 3 Distribution of papers over time (N = 23)

The papers addressed four primary MVP development phases: Planning, Requirements, Design and Development. Figure 4 presents the distribution of papers per MVP activity.

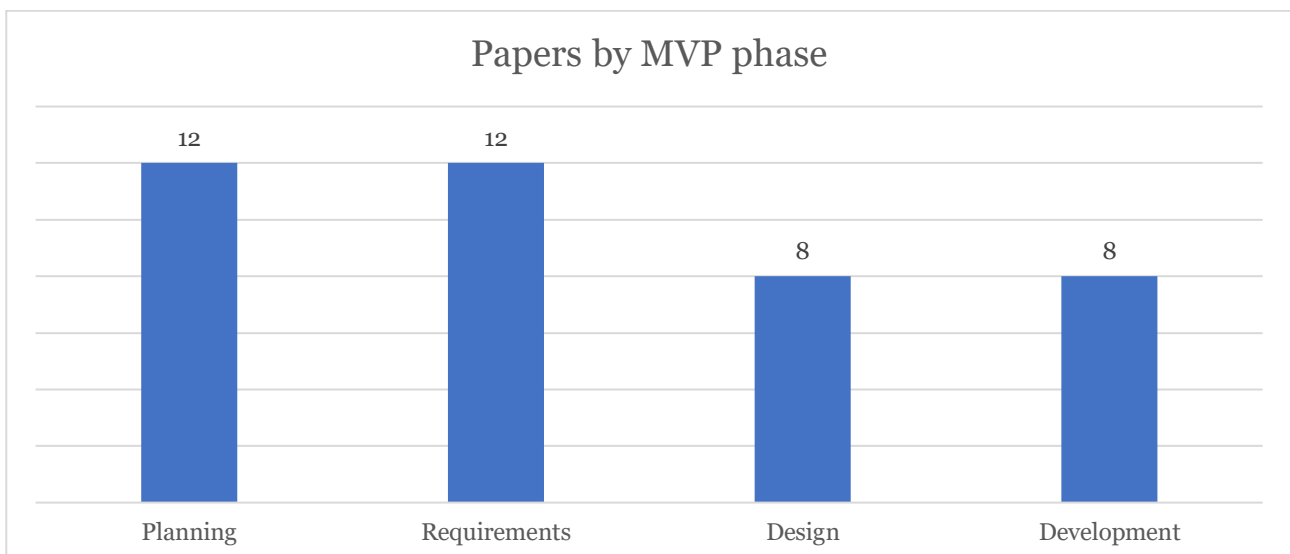


Figure 4 Distribution of selected papers across MVP development phases (N=23)

As we can observe in Figure 4, the requirements phase and planning phase received most attention in the literature with 12 papers each. The planning phase covered tasks like effort estimation, resource allocation, scheduling and risk assessment. The requirement phase addressed activities such as requirements extraction, analysis, prioritization and validation. The design and development phases were addressed in 8 papers, focusing on architecture design, prototyping, code generation and implementation support.

AI techniques were categorized in five groups: Natural Language Processing, Optimization Algorithms, Expert Systems, Deep Learning and Machine Learning. Figure 5 presents the distribution of these techniques across the literature.

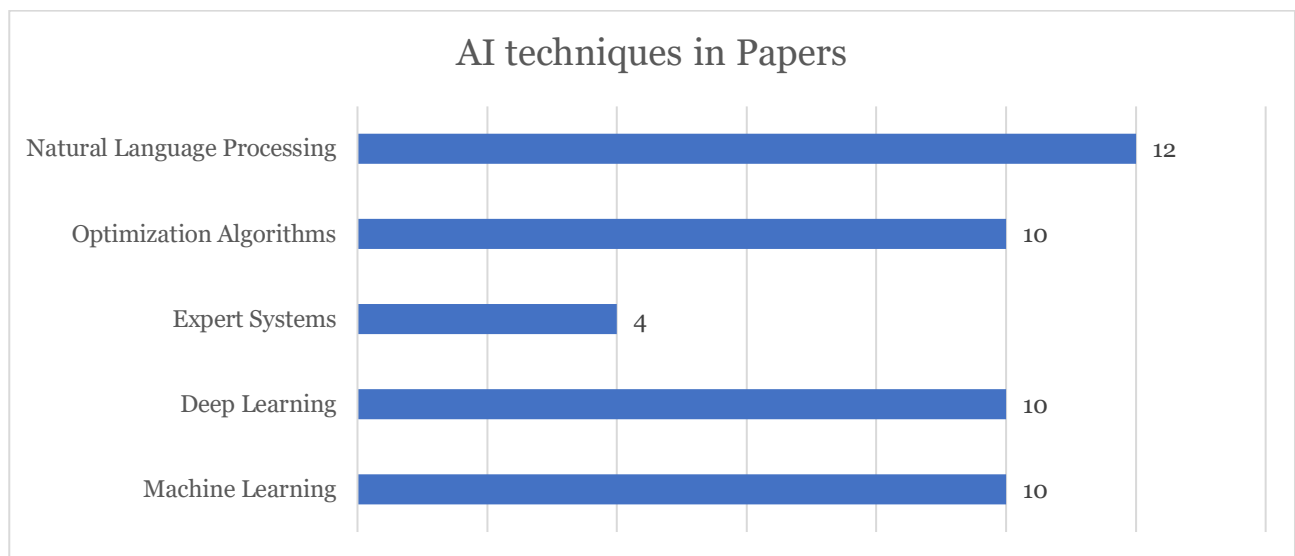


Figure 5 Distribution of AI techniques across selected papers (N=23)

As we can see from Figure 5 Natural Language Processing appeared as the most studied technology, appearing in 12 papers. This frequency reflects the substantial role of textual data analysis in software engineering activities, particularly in requirements engineering and documentation. Machine Learning, Deep Learning and Optimization Algorithms appeared each within 10 papers, while Expert Systems received less attentions, being addressed in only 4 papers.

## 4.2 AI techniques that Support MVP development Activities (RQ1)

This section addresses the first research question by examining how AI techniques support various MVP development activities across the software engineering lifecycle. As we can observe from Figure 6, the literature reveals that different AI techniques tend to align with specific MVP phases, depending on the nature of the tasks in each phase. The structure follows the phases presented in section 2.1.

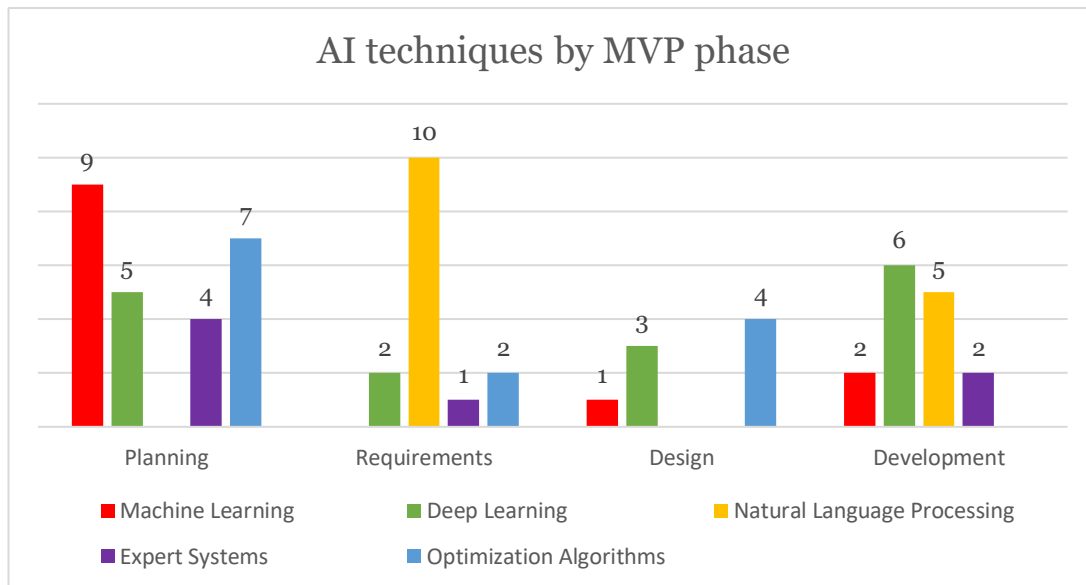


Figure 6 Mapping of AI techniques to MVP development phases

Machine Learning proved to be the strongest application in the Planning phase with 9 papers, but with limited presence in Development (2 papers) and Design (1 paper). Deep Learning showed more balanced distribution across all phases with a primary focus on Development (6 papers), followed by Planning (5 papers), Design (3 papers) and Requirements (2 papers).

Natural Language Processing presented clear specialization in the Requirements phase (10 papers), where textual data analysis is valuable, with secondary application in Development (5 papers). Expert Systems concentrated on Planning (4 papers) with limited presence in Development (2 papers) and Requirement (1 paper). Optimization Algorithms showed strong application in Planning (7 papers) and Design (4 papers) and moderate use in Requirement engineering (2 papers).

Mapping patterns reveal that certain AI techniques align naturally with specific MVP phases based on the nature of tasks in that activity. The Requirement phase is dominated by NLP, planning phase benefits from ML, DL, ES and OA, Development phase is mostly supported by DL and NLP capabilities, while Design phase utilizes OA and DL technology.

#### 4.2.1 AI Support in the Planning phase

The Planning phase benefits from various AI techniques, with Machine Learning and Optimization Algorithms displaying the strongest application. Studies show that AI techniques enhance effort estimation by tuning Machine Learning models specifically to the project data, thus supporting accurate planning and resource allocation (Xia et al., 2022). AI algorithms optimize scheduling, task allocation and predict project timelines and risks to improve MVP scoping and resource management (Barenkamp et al., 2020).

Machine Learning techniques, particularly Case Based Reasoning (CBR) and Classification and Regression Trees (CART), enable a more precise and adaptive effort estimation (Huang et al., 2015). Deep Learning based Neural Networks support accurate effort estimation in

the early phases of MVP development, mitigating the risk for under-budgeting or over-budgeting and streamlining scheduling (Sreekanth et al., 2023). The AI model learns from historical story-point data, providing consistent and accurate predictions, that help reducing estimation error in sprint planning (Choetkiertikul et al., 2019).

Optimization Algorithms process the complexity of project scheduling and resource allocation. Multi-objective memetic algorithms optimize scheduling goals such as cost, duration, and stakeholder satisfaction. OAs support dynamic scheduling in an MVP by enabling real-time adaptive rescheduling when the project state changes (Shen et al., 2018). Ensemble Machine Learning techniques, combining tools such as Bagging with Regression Trees or Multi-Layer Perception, provide more reliable approximations and improve the accuracy of software effort estimation (Minku & Yao, 2013).

In addition to the techniques mentioned above, Expert Systems are used to support the planning phase through rule-based decision-making. Peischl et al. depict an expert system that selects appropriate planning routines by referencing past project knowledge to identify key items for task definition (Barenkamp et al., 2020).

#### **4.2.2 AI Support in the Requirement phase**

Natural Language Processing dominates the Requirement phase, appearing in 10 papers of 12 that address this phase. This concentration reflects the nature of text-intensive activities of the Requirement phase. NLP techniques automate the analysis of natural language, detects uncertainties, identifies key concepts and maintains traceability, thus increasing accuracy and reducing manual effort (Zhao et al., 2021).

NLP automates the parsing of user stories by extracting the who, what, and why elements. This supports requirements analysis and artifact generation, reducing manual effort in defining MVP requirements (Raharjana et al., 2021). AI techniques automate several requirements-related tasks, including detecting unclear language, classifying requirements, identifying key domain concepts, and linking related requirements. This improves the accuracy and thoroughness of the requirements (Zhao et al., 2021). AI-powered automatic requirement smell detection provides immediate feedback on linguistic errors, such as ambiguities, loopholes and non-verifiable terms (Femmer et al., 2017).

NLPs enables the automated transformation of user stories to UML (Unified Modelling Language) Use Case diagrams, reducing manual labor and time to formalize requirements (Elallaoui et al., 2018). The requirement traceability is supported by combining multiple experts' opinion from mined software repositories, including commit messages and bug reports (Ali et al., 2013).

Optimization Algorithms complement NLP in requirements engineering with multi-objective approaches. Multi-objective Ant Colony Optimization (ACO) assists in automatically selecting optimal or near-optimal sets of requirements for MVP when resources, such as development time, are limited. This supports faster and more effective selection of the minimum set of features needed for an MVP release (del Sagrado et al., 2015). Pareto-based multi-objective optimization tools and weighted scalarization methods assist in balancing

trade-offs in an MVP, such as cost versus performance requirements. The tools support in choosing solutions that best fit the project's priorities (Chen & Li, 2023).

Knowledge-based systems (KBSs) support the requirement engineering by managing decisions related to requirements during the design process. Combined with NLP techniques, they improve the understanding of the problem domain and help in detecting incomplete or ambiguous requirement specifications (Shehab et al., 2020).

#### 4.2.3 AI Support in the Design phase

The design phase primarily utilizes Optimization Algorithms and Deep Learning. Evolutionary algorithms support MVP design and planning by automatically selecting feature sets that meet complex constraints while balancing multiple goals, such as cost and defect reduction (Xue et al., 2016).

Deep Learning tools, particularly Convolutional Neural Networks (CNNs) automate translating graphical UI (user interface) mock-ups into prototypes, supporting fast iteration UI design during the MVP development (Moran et al., 2020). CNNs learn to identify user interface elements and layouts from large datasets, allowing fast transformation of design files into working prototypes.

Genetic Algorithms assist in software architecture design by optimizing modularity, complexity, and other quality attributes (Ammar et al., 2012). Together, these AI techniques streamline the design phase, enabling a faster and more accurate MVP development.

#### 4.2.4 AI Support in the Development phase

During the development phase, the most utilized AI techniques are NLPs and DLs. CNN tools automate the conversion of UI mock-ups into working prototypes by learning to recognize interface components and layouts from large datasets (Moran et al., 2020).

NLP primarily uses methods such as grammatical dependency parsing with tools like the Stanford NLP parser to extract specific development tasks from free-form documentation, helping developers quickly understand and navigate large codebases (Treude et al., 2015). Machine Learning techniques assist in retrieving relevant semantic questions and ranking code snippets, using methods such as pairwise learning-to-rank. Deep Learning, using BERT transformer models for contextual embeddings, further improves detection of semantic similarity and enhances code search (Gao et al., 2023).

Expert Systems aid development through rule-based approaches that guide code generation and offer development support (Ammar et al., 2012). By encoding domain knowledge and best practices, they help developers make implementation decisions that align with MVP constraints.

<p><b>Summary of RQ1:</b> The mapping reveals that certain AI techniques align naturally with specific MVP activities, based on task characteristics. Requirement engineering with a focus on textual analysis is dominated by NLP extensively. Planning activities, involving effort estimation,</p>
---

scheduling and resource allocation benefits most from Machine Learning and Optimization Algorithms. During the design phase, Deep Learning and Optimization Algorithms were most effective for UI and architecture design, while the development phase was primarily supported by NLP, Deep Learning, and Expert Systems for extracting tasks, generating code, and converting mock-ups into working prototypes.

### 4.3 Trust and Accuracy of AI techniques (RQ2)

The second research question examines to which extent can the AI techniques be trusted in the development of an MVP.

#### 4.3.1 Accuracy Metrics and Performance

Study by Mustaqeem et al. (2023) demonstrate that different AI techniques reach high level of accuracy. Experimental results on a NASA repository shows that AI models received high precision (85%-91%), recall (88%-98%), F1-score (91%-94%) and accuracy (85%-88%) in software defect prediction. In the paper by Moran et al. (2020) CNN classifiers achieved a high average precision of 91% in graphical UI classification outperforming baseline approaches.

In requirement engineering, evaluation of user stories showed high precision and recall for NLP-based techniques: actors received a precision and recall score of 98%, while use cases and relationships showed precision of 87% and recall of 85% (Elallaoui et al., 2018). The accuracy of extracting development tasks from documentation with NLP tools reached over 90% when validated against manually noted benchmarks (Treude et al., 2015).

Effort estimation techniques demonstrate accuracy levels consistent with industrial standards. Optimized AI models applied to both waterfall and contemporary projects datasets received Magnitude of Relative Error (MRE) and Mean Magnitude of Relative Error (MMRE) values within the typical industrial range of 0.3-0.4 (Xia et al., 2022). Deep Learning based Neural Networks (NNs) reduce relative error, MMRE and Mean Absolute Relative Error (MARE) compared to existing methods (Sreekanth et al., 2023). In the paper by Choe-tkiertikul et al. (2019) story point estimation models leveraging Long-Deep Recurrent Neural Network (LD-RNN) technology consistently outperform common baselines including mean, median and random guessing in Mean Standardized Error (MAE) and Standardized Accuracy (SA) metrics.

Table 3 provides an overview of accuracy and performance of specific techniques, summarized by phase, technique, task and metrics used.

Phase	Technique	Task	Metrics	Source
Planning	ML, OA	Effort Estimation	MRE < 0,4	Xia et al., 2022
Planning	DL	Effort Estimation	RE ≈ 0,31 MRE ≈ 0,17 MMRE ≈ 0,19 MARE ≈ 0,28	Sreekanth et al., 2023

			PP $\approx$ 0,38	
Development	DL	Defect Prediction	Accuracy $<$ 0,9	Shehab et al., 2020
Planning, Requirement, Design	ML, DL, NLP, OA	Defect, Cost & Effort Prediction	Precision $\approx$ 0,89 Recall $\approx$ 0,93 F1 $\approx$ 0,93 Accuracy $\approx$ 0,87	Mustaqeem et al., 2023
Planning	ML	Effort Estimation	MMRE $\approx$ 0,17 PRED(25) $\approx$ 0,70	Mahmood et al., 2022
Requirement	NLP	User Story Extraction	Precision $\approx$ 0,75	Raharjana et al., 2021
Planning	ML, DL, OA	Cost Estimation	MBRE $\approx$ 3,90 PRED(25) $\approx$ 0,70 MdBRE $\approx$ 0,25	Huang et al., 2015
Design, Development	ML, DL	User Story to GUI	Precision $\approx$ 0,91	Moran et al., 2020
Requirement	NLP	User Story to UML	Precision $\approx$ [0,87;0,98] Recall $\approx$ [0,85;0,98]	Elallaoui et al., 2018
Development	NLP	Extracting Development Tasks	Accuracy $\approx$ 0,7	Treude et al., 2015
Planning	OA	Project Scheduling	HVR $\approx$ 0,88 IGD $\approx$ 0,12 Spacing $\approx$ 0,03 Spread $\approx$ 0,44	Shen et al., 2018
Planning, Design	OA	Feature Selection	HV $\approx$ 0,30 %Correctness $\approx$ 0,98	Xue et al., 2016
Development	ML, DL, NLP	Code Recommendation	Precision $\approx$ [0,46;0,59]	Gao et al., 2023
Planning	DL	Effort Estimation	MAE $\approx$ 2,09 SA $\approx$ 52,7	Choetkiertikul et al., 2019

Table 3 Summary of metrics performed

### 4.3.2 Trust and Limitations of AI techniques

Trust in AI techniques extends past accuracy and metrics. Trust depends on the data quality and the transparency of AI models. As discussed above, many AI models rely on past project data, historical data and training data. In a study conducted by Yang et al. (2020) imperfect training data affected the performance of a DL based classifier. In tools such as DNNs, the decision-making process is inherently obscure, as these models function like “black boxes,” resulting in low transparency (Yang et al., 2020). In another paper by Huang et al. (2015) the researchers state that preprocessing techniques may significantly affect the final prediction of AI models.

Despite promising accuracy levels, most of the studies performed were conducted in a laboratory setting with limited industrial validation. Zhao et al. (2021) noted that over 67% of the

studies related to NLP4SE (Natural Language Processing for Software Engineering) were conducted in a lab environment, while only a small percentage of 7% were conducted in an industrial setting. This gap limits the understanding of real-world applications of NLP techniques.

**Summary of RQ2.** The papers show that numerous AI techniques achieve high accuracy in MVP-related tasks, including UI classification, requirement analysis and effort estimation. Metrics such as precision, recall and F1-score consistently indicate strong performance. However, the reliability depends on data quality, model transparency and preprocessing methods. Imperfect data can reduce performance. In addition, most studies are conducted in lab settings rather than real industrial environments, leaving uncertainty about how these techniques generalize to real-world MVP development.

#### 4.4 Efficiency Gains from AI Techniques (RQ3)

The third research question addresses whether AI techniques increase the efficiency of developing an MVP. Table 4 summarizes key efficiency gains according to the literature.

Phase	Efficiency Type	Examples	Sources
Planning	Prediction & Estimation	Effort estimation, Resource allocation, Dynamic scheduling	
Requirements	Analysis & Automation	Requirements analysis, User story processing, Quality defect detection	(Ali et al., 2013; Ammar et al., 2012; Barenkamp et al., 2020; del Sagrado et al., 2015; Elallaoui et al., 2018; Femmer et al., 2017; Raharjana et al., 2021; Shehab et al., 2020; Yang et al., 2020; Zhao et al., 2021)
Design	Automation & Optimization	UI automation, Architecture optimization, Rapid prototyping	(Ammar et al., 2012; Barenkamp et al., 2020; Moran et al., 2020; Shehab et al., 2020; Yang et al., 2020)
Development	Code Assistance & Retrieval	Code generation, Code search, Documentation navigation	(Ammar et al., 2012; Barenkamp et al., 2020; Gao et al., 2023; Moran et al., 2020; Shehab et al., 2020; Treude et al., 2015; Yang et al., 2020)

Table 4 Summary of Efficiency Gains

##### 4.4.1 Automation of Repetitive Tasks

The automation of repetitive tasks is one of the key efficiency gains from using AI techniques. As stated by Lubars et al. “The reuse of experts design knowledge can play a significant role in improving the quality and efficiency of the software development process” (Yang et al., 2020). According to Shehab et al. (2020) AI-based tools can help in generating code, executing tests and analysis for code and constantly develop as the tools gather more information regarding the project.

AI tools reduce manual review time by automating NLP tasks such as defect detection, glossary extraction, and document classification (Zhao et al., 2021). They also decrease the effort required to analyze and manage user stories, enabling quicker agile iterations (Raharjana et al., 2021). Femmer et al. (2017) noted that the automatic detection of requirement smells increases the awareness of quality defects.

#### **4.4.2 Reducing Time in Development Cycles**

AI techniques accelerate development cycles by reducing time required for key activities. Automatic generation of UML Use Case diagrams from user stories significantly reduce the time and effort needed compared to manual modeling, enabling developers to focus on task that require human attention (Elallaoui et al., 2018).

Deep Learning tools enable faster and more accurate estimations thus allowing better timing and budgeting for lean development iterations (Sreekanth et al., 2023). More accurate estimates enable quicker spring planning and iteration, supporting lean MVP development (Choetkiertikul et al., 2019).

#### **4.4.3 Resource Optimization and Developer Productivity**

As discussed in Chapter 4.2, AI techniques play a significant role in optimizing resources and improving developer productivity. Many of the reviewed studies highlight strong applications of AI in the planning phase, particularly in effort estimation and resource allocation. By automating these tasks, AI reduces the time developers spend on activities that are repetitive or heavily clerical, allowing them to focus on work that cannot be automated.

Automation also improves overall productivity by streamlining documentation and communication processes. Tools that generate or classify documentation, summarize information, or extract key tasks from text help reduce manual workload and support clearer communication within development teams. This enhances collaboration and ensures that developers can dedicate more time to problem-solving and task that require human input.

Overall, AI-driven automation in planning, documentation, and communication contributes directly to more efficient resource use and increased developer productivity across MVP development.

<b>Summary of RQ3.</b> AI techniques automate repetitive tasks such as code generation, document classification, code analysis and executing tests, reducing manual review time and speeding up
---

workflows. In addition to these, AI shortens development cycles by generating models like UML diagrams and providing more accurate effort estimates that support lean planning. Beyond speeding up tasks, AI improves resource allocation and developer productivity. By handling repetitive work, AI allows developers to focus on tasks that need human supervision. These effects combined make MVP development faster, more streamlined and resource efficient.

## 5 Discussion

This chapter presents the findings of the literature review. The results indicate that AI techniques align well with lean methodologies and provide efficiency gains through task automation and estimation. The following sections discuss these findings in detail: Section 5.1 examines how AI supports the iterative nature of MVP development; Section 5.2 considers the real-world applicability of these tools; and Section 5.3 identifies gaps in the literature.

### 5.1 Supporting Iterative Development in MVP

The findings of this literature review reveal that AI techniques align with the iterative and lean nature of Minimum Viable Product development. MVP development, by definition, emphasizes rapid iteration, minimal resource investment and continuous learning through feedback. AI techniques serve as a powerful enabler of this iterative cycle.

However, the effectiveness of these techniques depends on the quality of data and historical data available within the organization. MVP development in startups or early-stage teams may lack historical project data, which could limit the capability data-driven AI approaches.

### 5.2 Real-World Validation

A significant limitation identified in the literature is the lack of industrial-scale validation. As mentioned in section 4.3.2, about 67% of studies focusing on NLP4RE were conducted in lab-setting, with only 7% conducted in actual industrial environments (Zhao et al., 2021). In addition, the studies examined in the other papers included in this literature review were also conducted in controlled environments. This gap raises important questions about the generalizability of AI techniques to real-world context.

Controlled conditions operate under clean, well-structured data and environments. In contrast, real MVP development occurs with imperfect data that is noisy, incomplete or inconsistent.

The performance metrics reported in the studies demonstrate high accuracy; however, these metrics do not guarantee equivalent performance in real-world circumstances.

### 5.3 Limited Coverage of Large Language Models and Generative AI

A notable gap in this literature review is the absence of research on Large Language Models and Generative AI in MVP development. This represents a significant gap given the rapid adoption of LLMs such as ChatGPT, GitHub Copilot and similar tools in software development practices since 2022. The current literature does not engage with these transformative technologies, limiting its applicability in present MVP development scenarios.

## 6 Conclusion

This thesis examined the intersection of Artificial Intelligence and Minimum Viable Product development. The primary objective was to assess how different AI techniques support the phases of MVP development and evaluate their accuracy and efficiency. To achieve this, a literature review was conducted using forward snowballing based on the study by Durrani et al. (2024). By synthesizing the findings, this thesis provides an overview of how AI can support MVP development, highlighting both the benefits and the practical challenges involved.

### 6.1 Key Findings

The literature demonstrates that AI techniques align with specific MVP phases based on task characteristics. Natural Language Processing dominates requirement engineering with text-intensive nature of user story analysis and requirement extraction. Machine Learning and Optimization Algorithms provide support in the planning phase, addressing effort estimation, scheduling and resource allocation. The design phase benefits from Deep Learning and OA tools, automating UI and architecture design. In the development phase, DL and NLP are utilized for code generation and task generation.

The reviewed techniques achieve high accuracy levels in MVP-related tasks. However, accuracy depends critically on data quality and data preprocessing. The accuracy tests were also conducted in a controlled environment, limiting confidence in a real-world setting. Deep Learning models and other similar complex models suffer from limited transparency. While accuracy metrics are promising, industrial scale validation remains unresolved.

The literature indicates that AI techniques can significantly improve efficiency across the whole lifecycle of MVP development by automating repetitive tasks and supporting resource optimization. These benefits allow for shorter development cycles and in turn allow developers to focus on higher-value activities and support lean iteration. However, the effectiveness of these gains depends on successful integration into team workflows, as adoption challenges and low-precision tools may reduce the overall benefits.

### 6.2 Limitations

Only 23 papers were selected for this review after applying the inclusion and exclusion criteria. This relatively small sample size may not capture all the relevant research in this specific area. In addition, the inclusion criteria requiring English-language articles and availability through Aalto Primo may have excluded relevant papers.

The dramatic rise of LLMs and generative AI tools post-dates most papers in this review. This presents a relevant gap given current industry adoption of these tools.

As noted in findings, most studies were conducted in a laboratory setting with controlled environment. This limits confidence in applying findings in a real-world MVP setting.

### **6.3 Future Research**

Based on the gaps identified in this review, several options for future research emerge.

Future work should prioritize conducting controlled studies with real SE teams in industry setting. In addition to accuracy metrics, these studies measure actual time savings, quality improvements and efficiency gains when AI tools are deployed in authentic MVP development context. Additionally, research should analyze the cost–benefit of integrating AI tools into SE teams, including implementation, training, and maintenance costs.

As Large Language Models and generative AI tools are becoming increasingly more prevalent, research should focus on how these tools can benefit the development of an MVP. Studies focusing on the transparency and explainability of these AI tools would further enhance their trustworthiness and practical applicability.

## References

- Alenezi, M., & Akour, M. (2025). AI-Driven Innovations in Software Engineering: A Review of Current Practices and Future Directions. *Applied Sciences*, *15*(3), Article 3. <https://doi.org/10.3390/app15031344>
- Ali, N., Guéhéneuc, Y.-G., & Antoniol, G. (2013). Trustrace: Mining Software Repositories to Improve the Accuracy of Requirement Traceability Links. *IEEE Transactions on Software Engineering*, *39*(5), 725–741. <https://doi.org/10.1109/TSE.2012.71>
- Alonso, S., Kalinowski, M., Ferreira, B., Barbosa, S. D. J., & Lopes, H. (2023). A systematic mapping study and practitioner insights on the use of software engineering practices to develop MVPs. *Information and Software Technology*, *156*, 107144. <https://doi.org/10.1016/j.infsof.2022.107144>
- Ammar, H., Abdelmoez, W. M., & Hamdi, M. (2012). Software Engineering Using Artificial Intelligence Techniques: Current State and Open Problems. *1st Taibah University International Conference on Computing and Information Technology*, *52*, 1–24.
- Barenkamp, M., Rebstadt, J., & Thomas, O. (2020). Applications of AI in classical software engineering. *AI Perspectives*, *2*(1), 1. <https://doi.org/10.1186/s42467-020-00005-4>
- Chen, T., & Li, M. (2023). *The Weights Can Be Harmful: Pareto Search versus Weighted Search in Multi-objective Search-based Software Engineering* | *ACM Transactions on Software Engineering and Methodology*. *32*(1), 1–40. <https://doi.org/10.1145/3514233>
- Choetkiertikul, M., Dam, H. K., Tran, T., Pham, T., Ghose, A., & Menzies, T. (2019). A Deep Learning Model for Estimating Story Points. *IEEE Transactions on Software Engineering*, *45*(7), 637–656. <https://doi.org/10.1109/TSE.2018.2792473>
- del Sagrado, J., del Águila, I. M., & Orellana, F. J. (2015). Multi-objective ant colony optimization for requirements selection. *Empirical Software Engineering*, *20*(3), 577–610. <https://doi.org/10.1007/s10664-013-9287-3>
- Durrani, U. K., Akpinar, M., Fatih Adak, M., Talha Kabakus, A., Maruf Öztürk, M., & Saleh, M. (2024). A Decade of Progress: A Systematic Literature Review on the Integration of AI in Software Engineering Phases

- and Activities (2013-2023). *IEEE Access*, 12, 171185–171204. <https://doi.org/10.1109/ACCESS.2024.3488904>
- Elallaoui, M., Nafil, K., & Touahni, R. (2018). Automatic Transformation of User Stories into UML Use Case Diagrams using NLP Techniques. *Procedia Computer Science*, 130, 42–49. <https://doi.org/10.1016/j.procs.2018.04.010>
- Femmer, H., Méndez Fernández, D., Wagner, S., & Eder, S. (2017). Rapid quality assurance with Requirements Smells. *Journal of Systems and Software*, 123, 190–213. <https://doi.org/10.1016/j.jss.2016.02.047>
- Gao, Z., Xia, X., Lo, D., Grundy, J., Zhang, X., & Xing, Z. (2023). I Know What You Are Searching for: Code Snippet Recommendation from Stack Overflow Posts | ACM Transactions on Software Engineering and Methodology. *ACM Transactions on Software Engineering and Methodology*, 32(3), 1–42. <https://doi.org/10.1145/3550150>
- Huang, J., Li, Y.-F., & Xie, M. (2015). An empirical analysis of data preprocessing for machine learning-based software cost estimation. *Information and Software Technology*, 67, 108–127. <https://doi.org/10.1016/j.infsof.2015.07.004>
- Lenarduzzi, V., & Taibi, D. (2016). *MVP Explained: A Systematic Mapping Study on the Definitions of Minimal Viable Product*. <https://doi.org/10.1109/SEAA.2016.56>
- Mahmood, Y., Kama, N., Azmi, A., Khan, A. S., & Ali, M. (2022). Software effort estimation accuracy prediction of machine learning techniques: A systematic performance evaluation. *Software: Practice and Experience*, 52(1), 39–65. <https://doi.org/10.1002/spe.3009>
- Martínez-Fernández, S., Bogner, J., Franch, X., Oriol, M., Siebert, J., Trendowicz, A., Vollmer, A. M., & Wagner, S. (2022). Software Engineering for AI-Based Systems: A Survey. *ACM Trans. Softw. Eng. Methodol.*, 31(2), 37e:1-37e:59. <https://doi.org/10.1145/3487043>
- Minku, L. L., & Yao, X. (2013). Ensembles and locality: Insight on improving software effort estimation. *Information and Software Technology*, 55(8), 1512–1528. <https://doi.org/10.1016/j.infsof.2012.09.012>

- Moran, K., Bernal-Cárdenas, C., Curcio, M., Bonett, R., & Poshyvanyk, D. (2020). Machine Learning-Based Prototyping of Graphical User Interfaces for Mobile Apps. *IEEE Transactions on Software Engineering*, 46(2), 196–221. <https://doi.org/10.1109/TSE.2018.2844788>
- Mustaqeem, M., Siddiqui, T., Ahmad Khan, N., & Kumar, D. (2023). In-Depth Analysis of Various Artificial Intelligence Techniques in Software Engineering: An Experimental Study. *Journal of Information Technology Management*, 15(3), 162–181. <https://doi.org/10.22059/jitm.2023.93632>
- Nguyen Duc, A., & Abrahamsson, P. (2016). *Minimum Viable Product or Multiple Facet Product? The Role of MVP in Software Startups* (p. 130). [https://doi.org/10.1007/978-3-319-33515-5\\_10](https://doi.org/10.1007/978-3-319-33515-5_10)
- Paić, G., & Serkin, L. (2025). The impact of artificial intelligence: From cognitive costs to global inequality. *The European Physical Journal Special Topics*. <https://doi.org/10.1140/epjs/s11734-025-01561-8>
- Raharjana, I. K., Siahaan, D., & Fatichah, C. (2021). User Stories and Natural Language Processing: A Systematic Literature Review. *IEEE Access*, 9, 53811–53826. <https://doi.org/10.1109/ACCESS.2021.3070606>
- Ries, E. (2011). *The lean startup How today's entrepreneurs use continuous innovation to create radically successful businesses*.
- Russell, S. J., & Norvig, P. (2021). *Artificial Intelligence A Modern Approach (4th ed.)*. Pearson.
- Schlicht, E. J. (2025). *Evaluating the Propensity of Generative AI for Producing Harmful Disinformation During the 2024 US Election Cycle* (No. arXiv:2411.06120; Version 8). arXiv. <https://doi.org/10.48550/arXiv.2411.06120>
- Shehab, M., Abualigah, L., Jarrah, M. I., Alomari, O. A., & Daoud, M. Sh. (2020). (AIAM2019) Artificial Intelligence in Software Engineering and inverse: Review. *International Journal of Computer Integrated Manufacturing*, 33, 1129–1144. <https://doi.org/10.1080/0951192X.2020.1780320>
- Shen, X.-N., Minku, L. L., Marturi, N., Guo, Y.-N., & Han, Y. (2018). A Q-learning-based memetic algorithm for multi-objective dynamic software project scheduling. *Information Sciences*, 428, 1–29. <https://doi.org/10.1016/j.ins.2017.10.041>
- Sidorkin, A. (2025). Environmental Impact of Generative AI: Carbon and Water Footprint. *AI-EDU Arxiv*. <https://doi.org/10.36851/ai-edu.vi.5448>

- Sofian, H., Yunus, N. A. M., & Ahmad, R. (2022). Systematic Mapping: Artificial Intelligence Techniques in Software Engineering. *IEEE Access*, 10, 51021–51040. <https://doi.org/10.1109/ACCESS.2022.3174115>
- Sreekanth, N., Rama Devi, J., Shukla, K. A., Mohanty, D. K., Srinivas, A., Rao, G. N., Alam, A., & Gupta, A. (2023). Evaluation of estimation in software development using deep learning-modified neural network. *Applied Nanoscience*, 13(3), 2405–2417. <https://doi.org/10.1007/s13204-021-02204-9>
- Thimmanna, A. V. N. S. S. (2024). Personalized Learning Paths: Adapting Education with AI-Driven Curriculum. *European Economic Letters (EEL)*, 14(1), 31–40. <https://doi.org/10.52783/eel.v14i1.993>
- Treude, C., Robillard, M. P., & Dagenais, B. (2015). Extracting Development Tasks to Navigate Software Documentation. *IEEE Transactions on Software Engineering*, 41(6), 565–581. <https://doi.org/10.1109/TSE.2014.2387172>
- Tripathi, N. (2019, November 19). *Initial minimum viable product development in software startups: A startup ecosystem perspective* [Väitöskirja]. Jultika.Oulu.Fi. <https://oulurepo.oulu.fi/handle/10024/36449>
- Xia, T., Shu, R., Shen, X., & Menzies, T. (2022). Sequential Model Optimization for Software Effort Estimation. *IEEE Transactions on Software Engineering*, 48(6), 1994–2009. <https://doi.org/10.1109/TSE.2020.3047072>
- Xue, Y., Zhong, J., Tan, T. H., Liu, Y., Cai, W., Chen, M., & Sun, J. (2016). IBED: Combining IBEA and DE for optimal feature selection in software product line engineering. *Applied Soft Computing*, 49, 1215–1231. <https://doi.org/10.1016/j.asoc.2016.07.040>
- Yang, Y., Xia, X., Lo, D., & Grundy, J. (2020). *A Survey on Deep Learning for Software Engineering* (No. arXiv:2011.14597). arXiv. <https://doi.org/10.48550/arXiv.2011.14597>
- Zhao, L., Alhoshan, W., Ferrari, A., Letsholo, K. J., Ajagbe, M. A., Chioasca, E.-V., & Batista-Navarro, R. T. (2021). Natural Language Processing for Requirements Engineering: A Systematic Mapping Study. *ACM Comput. Surv.*, 54(3), 55:1-55:41. <https://doi.org/10.1145/3444689>