

People's Democratic Republic of Algeria

Ministry of Higher Education and Scientific Research

Abderrahmane Mira University of Bejaia



Faculty of Exact Sciences

Department of Computer Science

Option: Advanced Information Systems

Introducing Agility in the Development Process of Intelligent Systems

Presented by:

MOH Mohamed & ZERGOUN Yasser

Board of examiners:

Supervisor: Mr. ACHROUFENE Achour / University of Béjaïa

President: Mr. BEDJOU Khaled / University of Béjaïa

Examiner: Ms. AIT HACENE Souhila / University of Béjaïa

Academic Year 2023-2024

Acknowledgments

We would like to express our deepest gratitude to Allah for giving us the strength, perseverance, and wisdom necessary to complete this work.

We would like to extend special thanks to our supervisor, Mr. Achour ACHROUFENE, whose expertise, patience, and wise counsel were invaluable in the completion of this thesis. His guidance illuminated our path throughout this process.

We would also like to express our appreciation to the members of the jury for their precious time and the great honor of evaluating this work.

A sincere thank you to our parents, family, and loved ones for their unwavering support, encouragement, and confidence in us. Their presence was our source of inspiration.

We would like to warmly thank our colleagues and friends who provided us with their help and encouragement throughout this journey.

Finally, we would like to express our gratitude to everyone who, near or far, contributed to the completion of this thesis.

Dedication

I would like to dedicate this thesis to several individuals who have been essential pillars throughout my academic and personal journey.

To my great father, your sacrifices and encouragement have shaped me into the person I am today. Thank you for always pushing me to reach for the stars and for instilling in me the values of dedication and perseverance, to my loving mother, my lovely fiancée, my beloved grandfather and grandmother. Your unwavering support, understanding, and belief in my abilities have been the driving force behind my academic pursuits. Thank you for always being there for me, providing a nurturing environment, and instilling in me the importance of education and perseverance.

To my supportive brothers Abderrahmane , Idris.

To my heart-kind sisters Meriem, Kawtar.

To my binôme, Moh Mohamed, I am incredibly grateful for your partnership and collaboration throughout this academic journey.

To my friends and roommates, especially Abdennour, Amine, Lyes, Zinedine, Anouar, for their friendship, camaraderie, and support throughout this academic journey. Your presence has made this experience all the more memorable and enjoyable.

Yasser

Dedication

I humbly dedicate this work to the people who are very dear to my heart:

To my parents, who have been a constant source of support, providing me with encouragement and financial backing that has enabled me to achieve my goals.

To my brothers, Younes, Aymen, and Abdelhak.

To my younger sister, Radjaa, who contributed to this thesis in her own small but significant way.

To my closest friend, Rima.

To my other friends and colleagues.

To my thesis partner, Yasser

And last but not least, to the best version of myself, who pushed through challenges and doubts, and whose commitment has carried me from the first year to this milestone moment.

Mohamed

Table of Contents

General introduction	1
Chapter 1 Intelligent Systems and Agile Development	3
Introduction	4
1.1 Intelligent system	4
1.2 What defines artificial intelligent?	5
1.3 What is an intelligent system?	5
1.4 Characteristics and proprieties of intelligent systems	6
1.4.1 Agents	6
1.5 Environment	7
1.6 Components of an intelligent system	8
1.7 Functioning of AI based systems	9
1.8 Classification of intelligent systems	10
1.8.1 Based on the functionalities of the system	10
1.8.2 Based on the level of autonomy	11
1.9 Applications and examples of AI-based systems	12
1.10 Development of an intelligent system (life cycle)	13
1.11 Software engineering methodologies	15
1.11.1 Software development	15
1.11.2 Agile Methods	19
1.11.3 Advantages of Agile Software Development	24
1.11.4 Disadvantages of Agile Software Development	24

Table of Contents

1.11.5 Comparing agile and traditional methodologies	25
Conclusion	27
Chapter 2 AI Systems Development Methodologies: State-of-the-Art	28
Introduction	29
2.1 Related works	32
2.1.1 Review and survey papers	32
2.1.2 Case study papers	34
2.1.3 Original papers	41
2.2 Comparative table	45
2.3 Summary	47
Conclusion	48
Chapter 3 ScrumAI: Adapting Scrum Agile Methodology for AI Development	49
Introduction	50
3.1 Problematic	50
3.2 Scrum-IA approach	51
3.2.1 Model cycle	53
3.2.2 Functioning of ScrumAI	53
3.3 Addressing AI development challenges with ScrumAI	57
3.3.1 Specifications for intelligent systems	57
3.3.2 Addressing the challenges	57
Conclusion	58
Chapter 4 ScrumAI in Action: A Real-World Case Study	59
Introduction	60

Table of Contents

4.1 Business need	60
4.2 The solution	61
4.3 Case study	61
4.3.1 Sprint zero	61
4.3.2 Functional requirements	62
4.3.3 Non-functional requirements	62
4.3.4 Roles	63
4.3.5 Product backlog	64
4.3.6 Tools and technology	65
4.4 Sprint 1	69
4.4.1 Development	70
4.5 Sprint 2	79
4.5.1 Development	80
4.6 Evaluating ScrumAI	86
Conclusion	88
General conclusion	89
References	91

Figures and Tables

Figure 1 - The component of an AI system	9
Figure 2 - The development cycle of AI system [17]	13
Figure 3 - The 12 agile principles	18
Figure 4 - Overview of XP [25]	20
Figure 5 - A hump chart that illustrates RUP architecture [26]	21
Figure 6 - Scrum overview for Agile Software Development [28]	23
Figure 7 - The Cross-Industry Standard Process for Data Mining (CRISP-DM) [31].	35
Figure 8 - The Team Data Science Process TDSP [31].	36
Figure 9 - The Microsoft model described by Amershi et al [31].	36
Figure 10 - Refined CRISP-DM model. Additions in red, with bold text [31].	37
Figure 11 - AMOD workflow [37].	41
Figure 12 - Q-Model workflow [38].	42
Figure 13 - Agile workflow for AI applications [39].	43
Figure 14 - ScrumOntoBDD workflow [40].	44
Figure 15 – The workflow of ScrumAI	52
Figure 16 - Detailed workflow of ScrumAI	56
Figure 17 - Code snippet using Tkinter to create user interface	66
Figure 18 - Code snippet using pandas to load a dataset	67
Figure 19 - code snippet using matplotlib to display graphs	68
Figure 20 - Code snippet from kaggle environment to train the machine learning model	69
Figure 21 - Lag features	73
Figure 22 - Use case diagram (sprint 1)	76

Figures and tables

Figure 23 - Model/Data version control example	78
Figure 24 - Forecasting interface	79
Figure 25 - Use case diagram (sprint2)	81
Figure 26 - Most selling product families	82
Figure 27 - Most selling stores	82
Figure 28 - Dashboard interface 1	85
Figure 29 – Dashboard interface 2	85
Table 1 - Forms of the environment [4]	7
Table 2 - Agile VS Traditional [18]	27
Table 3 - Paper classification	31
Table 4 - Paper’s results [36]	33
Table 5 - Comparative table	46
Table 6 - Distribution of roles	64
Table 7 - Product backlog	65
Table 8 - Sprint 01 backlog	70
Table 9 - Evaluation metrics (model 1)	74
Table 10 - Model documentation (sprint 1)	77
Table 11 - Sprint 02 backlog	80
Table 12 - Evaluation metrics (model 2) and a comparison between the two models	83
Table 13 - Model documentation (sprint 2)	86

Abbreviation list

AGM: Agile Methodologies.

AI: Artificial Intelligence.

BDD: Behavior-Driven Development.

CNN: Convolutional Neural Networks.

CRISP-DM: Cross-Industry Standard Process for Data Mining.

Dev-ops: Development Operations.

GAN: Generative Adversarial Networks.

MEA: Mean Absolute Error.

MLops: Machine Learning Operations.

NFR: Non-Functional Requirements.

PO: Product Owner.

RMSE: Root Mean Squared Error.

RNN: Recurrent Neural Networks.

RUP: Rational Unified Process.

SE: Software Engineering.

SDLC: Software Development Life Cycle.

TDM: Traditional Methods.

TDSP: Team Data Science Process.

General introduction

Artificial Intelligence (AI) has revolutionized various aspects of our lives, transforming the way we interact with technology and solve complex problems. From virtual assistants and personalized recommendations to autonomous vehicles and intelligent medical diagnosis, AI-powered systems are becoming increasingly ubiquitous. As AI continues to gain momentum, the development of these systems demands a structured methodology to ensure their efficient creation and deployment.

The development of AI systems poses unique challenges due to their distinct characteristics and components, such as sensors, reasoning engines, and machine learning algorithms. These complexities necessitate a tailored approach to address the specifications of AI systems, which can vary significantly from usual software development. Indeed, Waterfall and Agile approaches offer many advantages for software development, but fail to consider the specifications of AI system. For instance, developing a simple expert system requires collecting knowledge from a domain expert and building an inference engine to deduce knowledge from the knowledge base. It is also necessary to build models in systems based on machine learning techniques. These steps are not explicit in the usual development methodologies. Therefore, a well-defined methodology is essential to handle the intricacies of AI systems and ensure their successful development.

In response to these challenges, this thesis proposes an adapted methodology by extending the Scrum agile framework, called ScrumAI, to accommodate the unique requirements of AI system development. Agile methods are widely adopted by developers for their flexibility and acceptance of change during development. In particular, the Scrum method is widely used for its simplicity and practicality. ScrumAI aims to tackle the complexities and specifications of AI systems by improving the development process, involving the domain expert and cognitive scientist in AI sprints and adding new AI application practices while retaining the essence of Scrum.

The structure of this thesis is as follows:

- Chapter 1 will delve into the specifications and characteristics of AI systems, as well as their development process, and examine different development software methodologies, with a focus on agile frameworks.
- Chapter 2 will review the state of the art in development methodologies for AI applications, discussing and summarizing 13 relevant papers.

General introduction

- Chapter 3 will present the ScrumAI methodology in detail, outlining its workflow and how it addresses the challenges identified in Chapter 2.
- Finally, in order to validate the ScrumAI approach, Chapter 4 will provide a case study example to evaluate and discuss any potential weaknesses in the methodology. We will develop a real intelligent decision support system based on machine learning techniques that can predict store sales and display intelligent visual representations.

Chapter 1

Intelligent Systems and Agile

Development

Introduction

Artificial Intelligence (AI) has found its way into various industries and aspects of our lives. From virtual assistants and personalized recommendations to autonomous vehicles and intelligent medical diagnosis, AI-powered systems are transforming the way we interact with technology and solve complex problems.

Just like traditional applications, the creation of AI-powered systems also demands a structured methodology, and often involves the integration of special techniques and roles.

In this chapter, we will be delving into the specifications of AI applications, including their characteristics, components, and development life cycle. Furthermore, we will classify the different AI applications and provide real-world examples.

Additionally, we will explore the various types of development methodologies that exist. This will include a discussion on both traditional and agile methodologies, as well as a comparison of their advantages and disadvantages. We will also highlight the relevance of these methodologies in the context of intelligent system development.

By exploring these aspects, this chapter aims to provide a comprehensive understanding of the diverse applications of artificial intelligence and the methodologies employed in the development software applications in general.

1.1 Intelligent system

According to Oxford English Dictionary¹, Intelligence is the ability to acquire and apply knowledge and skills. What about artificial intelligence? What is an intelligent system?

¹ Compact Oxford English Dictionary, 2006

1.2 What defines artificial intelligent?

Artificial intelligence has seen many definitions across time; we refer to the most popular definitions [1]:

Artificial intelligence is any computer that passes the Turing test [2]. If a human being cannot differentiate between a computer and another human being, then the machine is considered intelligent.

Artificial intelligence is the science and engineering of making intelligent machines [3], allowing the machine system to achieve goals in a specific environment that usually requires a human to do it.

In other words, artificial intelligence is giving the software system or the machine the ability to act on its own and choose the right action in an intelligent way to reach a certain goal with high performance.

1.3 What is an intelligent system?

The concept of intelligent system has emerged in information technology as a type of system derived from successful applications of artificial intelligence [4]. Broadly speaking an intelligent system is often defined as a machine or software program designed to perform a useful task for human beings, such as recommending shopping, diagnosing diseases or driving a car. We also found several definitions in the literature, depending on the point of view of the authors.

Based on [5], intelligent systems, or AI-based systems, are software systems with functionalities enabled by at least one AI component (e.g., image or speech recognition, prediction or classification, and automating). According to [6], AI-based systems include any systems that use artificial intelligence algorithms (statistical machine learning algorithms, rule-based algorithms). As for [4], an AI-based system represents an agent that can analyze its environment, act on it, interact with other agents, and perform rational behavior. An intelligent system is an intelligent agent [7]. It is a software system that detects its environment through sensors and acts upon that environment through actuators by selecting an action that is expected to maximize a performance measure.

1.4 Characteristics and proprieties of intelligent systems

It can be challenging to define intelligent systems; these definitions describe possible characteristics that an intelligent system might have.

There are many types of AI-based systems with different use cases and goals; each uses a different approach, technology, and algorithm; therefore, it is necessary to explore all the different aspects and properties of intelligent systems to fully understand what AI-based systems are [1], [6]. Here, we rely on definitions based on the agent concept to describe some interesting properties.

1.4.1 Agents

Represent human users or other intelligent systems. An intelligent system agent can act autonomously in the environment to perform a specific task by taking its own decisions, or it can also play an advisor role for a human user agent, suggesting possible actions to take with the required information to help decide. However, the human user will make the final decision.

There are three ways of autonomous acting [4], [5]:

- **Reactive Decision:** This is the simplest form of acting. It includes reacting to changes with simple decisions and following a set of conditions. For example, an intelligent system for temperature control will turn on the air conditioner if the temperature of the room rises above a certain degree (condition).
- **Goal Reasoning:** This form of acting demands a certain level of reasoning in order to determine the set of actions required to reach a specific goal. This reasoning takes into account the right order and timing as well as the right values for the actions while having a memory to keep track of the states. For instance, a cleaning robot keeps track of battery level, places already visited and cleaned, and the charging location.
- **Actions reasoning:** In addition to goal reasoning, it has the ability to consider multiple sequences of actions, multiple paths, and their consequences to achieve the goal with the best performance and minimum loss.

1.5 Environment

The environments represent the space where the intelligent systems can function and act. It can take multiple forms and properties. Table 1 expresses the different forms of the environment.

ENVIRONMENT PROPERTY	DESCRIPTION
STATIC (OR DYNAMIC)	The environment does not change (or changes) while an agent is making a decision.
DISCRETE (OR CONTINUOUS)	The observed state of the environment, time or actions are discrete (or continuous).
FULLY- OBSERVABLE (OR PARTIALLY- OBSERVABLE)	Sensors detect (or do not detect) all aspects that are relevant to the choice of action.
DETERMINISTIC (OR STOCHASTIC)	The next state of the environment is (or it is not) completely determined by the current state and the action.
EPISODIC (OR SEQUENTIAL)	Actions do not have influence (or they have influence) on future actions.
KNOWN (OR UNKNOWN)	The outcomes for all actions are known (or they are not known) by the agent in advance.

Table 1 - Forms of the environment [4]

1.6 Components of an intelligent system

Intelligent systems are composed of various interconnected components that work together to process information and execute tasks (see Figure 1). Each component plays a crucial role in ensuring the system operates effectively and efficiently, whether it's capturing data from the environment or making complex decisions based on learned knowledge [4].

Sensors: Sensors capture data from the environment, such as text, images, sounds, temperature, motion, or other forms of input. These inputs serve as the system's perception of the world.

Reasoning and Inference Engine: The reasoning engine is the intelligent part of the system; it is responsible for the thinking processes. It takes data as input from sensors or existing datasets and uses the knowledge base to derive conclusions, make predictions, or solve problems. It applies several techniques and algorithms to interpret the data and generate insights. There are two types of reasoning:

- Rule-based reasoning, it applies simple if and else algorithms or a more complex rule-based algorithms such as forward chaining or backward chaining
- Mathematical-based reasoning, it applies machine learning models such as linear regression, decision trees, neural networks, and support vector machines.

Knowledge Base: The knowledge base contains domain-specific information that the intelligent system uses to make decisions or perform tasks. This knowledge can be represented in various forms, including per-defined rules, ontologies, or databases that have been verified by experts in the domain.

Actuators: actuators are components that enable the system to interact with its environment by producing physical actions or outputs. These actions can include controlling motors, adjusting parameters, returning information or displaying an image, or communicating with other systems.

User Interface: The user interface provides a way for users to interact with the intelligent system, presenting information in a human-readable format and allowing users to input commands or preferences.

The components of AI systems can vary significantly from one system to another. While some AI systems might encompass all the components, others might include only a few of these elements. The specific components used depend on the system's purpose, complexity, and application requirements.

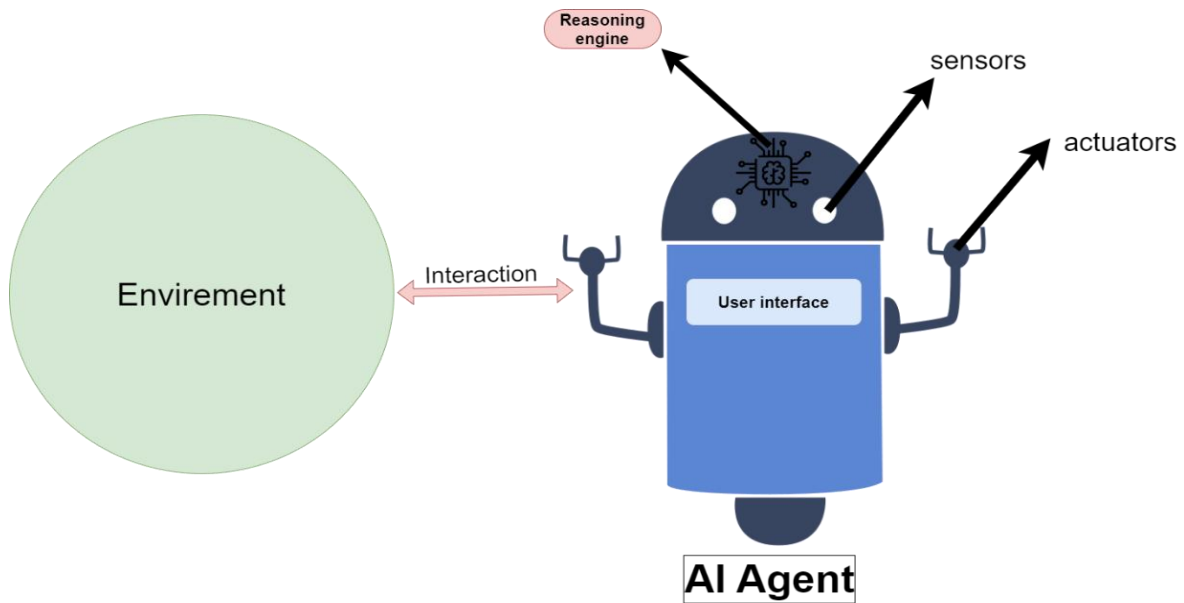


Figure 1 - The component of an AI system

1.7 Functioning of AI based systems

The functioning of an AI-based system may vary from one system to another, depending on the type of system and its purpose. However, AI-based systems have a number of common functionalities, especially in environment analysis and data processing.

Starting with environment analysis, it can be decomposed into three functionalities:

- **Data Acquisition:** The objective of this function is to collect the necessary information from either existing data or through the different sensors. The data may take various forms (image, temperature, pressure, text, light, etc.).

- World modeling or data processing: The system will process and generate a formal representation (machine-readable) of the data collected. For example, a self-driving car will try to simulate the world around it via the different information acquired from the different sensors (camera, GPS, etc.).
- Situation analysis, or understanding the environment and other external agents: It may include using predefined facts and verified beliefs that match the current situation. For example, an intelligent system for medical diagnosis will analyze the patient's symptoms by using already proven knowledge that relates or matches the input symptoms to an infection or a disease. Understanding the environment may also require calculating and assigning some values, such as the distance or time cost of some action.

After this environment analysis, the intelligent system can then take action in the environment.

Acting in the environment happens after a certain level of reasoning, which takes place in the reasoning engine or a statistical model. Executing actions using the different actuators (action component) occurs after receiving the exact sequence of commands and values for the controllers, for example, the action of slowing down in a self-driving car will be translated to a command for the “brakes” actuator, and a speed goal will be set.

Finally the intelligent system can also Communicate with other agents by sending or receiving different kinds of data (text, images, audio, values, etc.), which can be done through Wi-Fi, Bluetooth, or cables.

1.8 Classification of intelligent systems

Intelligent systems can be classified in several ways. In the following, we have chosen to focus on two key classifications that highlight the different functionalities of these systems and their autonomy hierarchy.

1.8.1 Based on the functionalities of the system

Recommendation systems such as search engines, YouTube recommendations, and Ali Express recommendations use machine-learning algorithms and large amounts of data gathered about user interactions, including impressions, clicks, likes, and purchases. They are capable systems of understanding the preferences, previous decisions, and characteristics of users [8].

Decision support systems, such as voice assistants, chatbots, or specialized systems within an enterprise, are intelligently trained programs that use data and knowledge from different sources. After analyzing the different data, they can provide a predictive decision with the necessary information to support and aid the decision-making process. This information can be in the form of graphs, models, tables, figures, or text [9].

Expert systems, such as image analysis software, speech and face recognition systems, and medical diagnostic systems, are computer programs that use artificial intelligence methods and algorithms (such as forward or backward chaining) to solve problems within a specialized domain that ordinarily requires human expertise; in other words, they simulate human expertise [10].

Form recognition systems are capable of recognizing forms, images, objects, sounds, characters, etc. from noise data that can be pixels or sound waves. An example of this system is face detection, which is found in smart phones for security, or audio-to-text applications that convert audio sound to text speech.

Content generation systems, the kind of application that is dominating this latest period, have the ability to interact with users, understand the natural language of users in different languages, and generate human-like content from scratch. They can generate images, text, audio, video, or even programming code. The most famous examples of these systems are “ChatGPT” for text and code and “DALL.E 2” for image generation [11].

Game AI: “IBM’s Deep Blue computer program once defeated the world champion in a chess match, proving the use of AI within the game industry” [12]. Games today include very intelligent systems that make games more entertaining and similar to real life, for example, giving a game object the ability to have a certain behavior, giving them a certain goal, such as attacking the player, and making them understand the game environment [13].

Note that an intelligent system can be classified into one or many classes at the same time. For example, the ChatGPT chatbot can be classified into content generation systems as well as decision support systems.

1.8.2 Based on the level of autonomy

According to [8], there exist three levels of autonomy:

Fully automatic systems: They are AI-based systems programmed to work automatically without any human interference. For example, a cleaning robot.

Semi-automatic systems: They represent AI-based systems that can function with a certain level of autonomy. They require an external agent or human user interference at some point to guarantee a proper functioning system. An example of this system is “self-driving cars.”

Interactive systems: These systems require a constant interacting user to work. This interaction is in the form of some text input, voice command, or other kinds of inputs, for instance, the Chabot systems like “ChatGpt” or voice assistance systems such as “Google Assistance.”

1.9 Applications and examples of AI-based systems

Artificial intelligence applications have spread rapidly in recent times, affecting many fields, including healthcare, finance, education, and factories [14], [15], [16].

In healthcare, for example, diagnostic assistance systems can help doctors analyze and detect diseases more accurately. Another example is recommending possible treatments based on the patient's symptoms.

In finance, there are intelligent trading systems that can autonomously trade money by analyzing market data and values based on predefined algorithms. There are also recommendation systems that suggest products, services, or videos based on user preferences and sentiment analysis.

In transportation, self-driving cars powered by AI can navigate roads safely and efficiently. Traffic analysis systems can find the optimal routing from point A to point B, similar to a "Google Maps" navigation.

In manufacturing, factories use intelligent maintenance systems that can detect anomalies and potential failures by monitoring and analyzing real-time sensor data on measures like pressure and temperature. Factories also deploy intelligent quality control systems that can inspect products and identify any defects.

In security, intelligent systems can be integrated with sensors like cameras and heat detectors to monitor for movement or perform facial recognition. In cybersecurity, AI-based systems are used to detect and filter out fraudulent emails.

In education, there are intelligent systems that can automatically grade assignments and provide feedback to students.

1.10 Development of an intelligent system (life cycle)

The life cycle of an intelligent system passes through a number of steps that might change slightly depending on the type of system. In the following case, we will explore a model-based intelligent system since it constitutes the majority of AI systems.

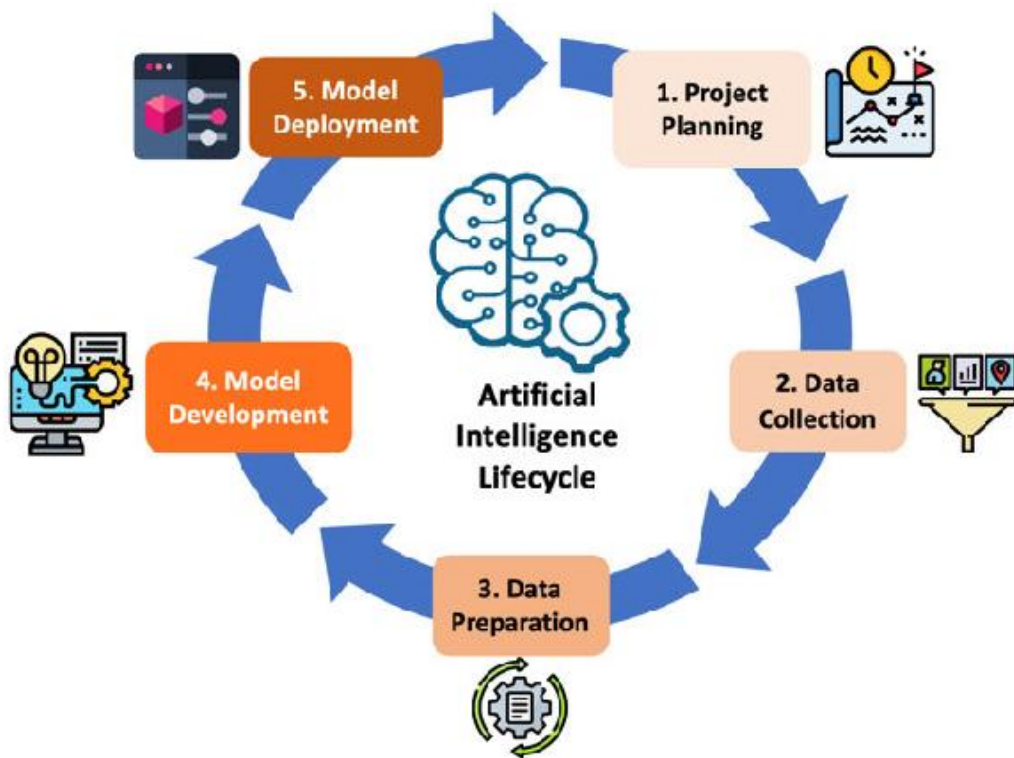


Figure 2 - The development cycle of AI system [17]

As shown in Figure 2 the life cycle of an AI system includes the following steps:

Setting the goal: this includes identifying the system requirements and functionalities, identifying the users, studying the market, and setting the key performance indicators.

Planning the development process: this includes calculating the cost and time, distributing the tasks, adopting a development methodology, selecting the tools and technology for the development process.

Collect and prepare the data: this phase requires an understanding of the domain to collect the right information and prepare high-quality data. This step includes two main activities:

- Data cleaning: it is used to remove inaccuracies and missing values. Common tools for that are OpenRafine and RapidMiner.
- Data pre-processing: it involves data transformation, normalization, and eliminating features and attributes. Common tools for that are Python math libraries such as Skit-learn.

Designing and building the model: it represents the implementation phase it includes the following steps:

Specifying the learning mechanism from the three main approaches:

- Supervised learning: the data used for training is already labeled, and the model has access to the correct answer, most commonly used for accurate prediction and recognition.
- Unsupervised learning: the data used for training is unlabeled and best used for classifying and clustering.
- Reinforcement learning: the model learns by rewarding it for the correct result (learning through trial and error).

Specifying the model architecture: this step is usually for complex systems using deep learning. Some of the top architectures are convolutional neural networks (CNN), recurrent neural networks (RNN), and generative adversarial networks (GAN).

Testing and Deploying:

- Conduct thorough testing to ensure the model's performance and reliability.
- Deploy the model to the production environment for real-world use.

In intelligent system development it is important to get the difference between software and hardware intelligence. Indeed, software intelligence refers to intelligence that is implemented and executed with a software program like a web application or a desktop application, moving from digitalization to AI integration, whereas hardware intelligence refers to intelligence being embedded into the hardware component, giving it the capacity to be and act intelligent. These involve special hardware designed to perform tasks such as moving parts.

Thus, hardware intelligence represents the container that holds software intelligence. In our case, we will be focusing on software intelligence and present software engineering methodologies in the next section.

1.11 Software engineering methodologies

This section provides an overview of software development methods, including traditional approaches like Waterfall and iterative methods like the Spiral model. It also introduces agile methodologies such as Scrum, Kanban, and Extreme Programming, highlighting their relevance in the context of intelligent system development.

1.11.1 Software development

Software development is an organized process that thrives on delivering products in faster, better, and cheaper ways. There have been many studies and suggestions for improving the development process.

The software development life cycle (SDLC) is the most important element in software development. It depicts the necessary phases in software development. SDLC is the process of building or maintaining software systems. Typically, it includes various phases, from preliminary development analysis to post-development software testing and evaluation. It also consists of the models and methodologies that development teams use to develop the software systems and the methodologies form the framework for planning and controlling the entire development process.

Currently, there are two SDLC methodologies that are utilized by most system developers, namely traditional development and agile development [18].

1.11.1.1 Traditional Software Development

Software methods such as Waterfall method, V-model and RUP are known as traditional software development methods and are among the heavy methods. These methods are based on processes such as requirements definition, solution design, testing and implementation. Traditional software development methods require a well-defined requirements at the beginning of the project.

Process

According to [18], there are four phases that are characteristic of the traditional software development method.

The first step is to determine the requirements of the project and determine the time required to implement the various stages of development, while trying to predict the problems that will arise in the project.

The next step is the research and design process, where the technical equipment is created in the form of diagrams. These outline potential problems that will be encountered as the project progresses and provide developers with a road map to work towards implementation.

A stage of development where code is released until some goal is achieved. Development is often divided into sub-tasks and divided into different groups based on capabilities.

The testing step often follows the development process to ensure that early problems are solved. When the project is finished and the manufacturer is close to meeting the requirements, the client becomes part of the test and feedback process and the project is delivered to the client after his full satisfaction.

Disadvantages of Traditional Software Development

The traditional software development methods are dependent on a set of predetermined processes and on-going documentation, which is written as the work progresses and guides further development. The success of a project which is approached in this way relies on knowing all of the requirements before development begins, which means that implementing change during the development life cycle can be somewhat problematic [19].

The disadvantages of traditional methods arise from various factors such as inflexibility, limited customers' involvement, late feedback and testing, high risk of project failure, long delivery cycles, lack of visibility and control, limited room for experimentation, and difficulty in managing complex projects [20].

1.11.1.2 Agile Software Development

The term agile means 'moving fast'. The agile methodology is a simple way of creating software. The agile model was proposed in 2001 by a group of software developers who came together in Utah to explore new and better ways of developing software, believing that each project should be approached differently and that existing methods should be adapted to best meet the needs of the project. In agile development, instead of a single large model implemented in the standard SDLC, the development lifecycle is broken down into smaller pieces called "increments" or "iterations", and each of these steps has an impact on each of the standard development phases. After each iteration, practical steps are taken and working software is delivered. Each building gains more character; The final building has everything the customer needs [21].

The success of a project requires regular and frequent feedback from the client. Thus, the customer work closely with the development team to provide continuous feedback that ensures better control of the project [22].

When planning, it is necessary to ensure that the schedule is flexible and adaptable to changes that may occur in the context, technologies, and specifications. Indeed, it is difficult to think of all the features you would like to have from the beginning, and it is very probable that the customer will modify its requirements once he sees an early version of the operating system.

Core values

The Agile model emphasize four core values [23]:

- Individuals and interactions over processes and tools
- Working software over comprehensive documentation
- Customer collaboration over contract negotiation
- Responding to change over following a plan

Principles

The agile methodology consists of several principles represented in Figure 3 [24]:



Figure 3 - The 12 agile principles

- 1** Our top priority is customer satisfaction through early and continuous delivery of valuable software: the focus is always on customer satisfaction.
- 2** The flexibility of agile methods make it accepting the changes even if the software was under development.
- 3** The goal of iterating is to deliver small additions at short, regular intervals. The length of the cycle usually varies from 1 to 3 weeks, with an extension given at the end of each cycle. This allows the customer to receive small orders in a short time, unlike the traditional method where large orders are delivered at the end of the project.
- 4** Frequent communication between developers and partners is preferred. Because it helps, the development team understand all the changes the customer needs and quickly integrate these changes while building small features.
- 5** Motivating developers in agile teams and thus creating an organized team is one of the most important aspects of agile.
- 6** The most effective and efficient way to communicate information in teams and during development is face-to-face communication.
- 7** Functional software is the most important measure of progress: The first sign of progress is functional software.

8 Development teams need to be able to deliver increasing value repeatedly after each iteration.

9 Increasing efficiency by maintaining a focus on design and technology is an important factor.

10 Encourages the team members to prioritize simplicity and avoid unnecessary complexity in all aspects of the software.

11 A better architecture, requirements and designs comes from self-organizing teams, which increase the efficiency of the team.

12 At the end of each iteration, there are 'Iteration Review' and 'Iteration Retrospective' events. These help teams evaluate their work during iteration and identify many things that are not going so well.

1.11.2 Agile Methods

Agile methods focus more on IT project management. They are based on iteration to the development according to customer changes. In particular, it allows all employees and customers to participate in the development of the software.

For more than a decade, there has been an increase in the number of fast methods available, including technologies and techniques specific to software development. Agile methods are a subset of systematic and adaptive methods and are based on iterative processes of improvement and development [22].

These methods can effectively meet customer expectations and improve employee skills in a short period. These methods are productivity, competitive advantage, and both customer and supplier sides. Currently, there are many agile methods in use; the most popular methods are described briefly in this section.

1.11.2.1 eXtrem Programming

The main goal is to reduce the cost of change. Traditional methods involve defining and determining the requirements at the beginning, followed by increasing costs as changes are made. XP strives to make the program more flexible and open to change by incorporating core values, principles and practices.

The client (project owner) manages the project, while the other team members release the initial version of the software into the stream, and new versions are released one after another at a steady pace for critical feedback on the progress of the project. A group organizes itself to achieve its goals by promoting cooperation among its members. The team implements automated tests for every feature it develops, ensuring a high level of robustness [25].

As the Figure 4 shows the process of XP method

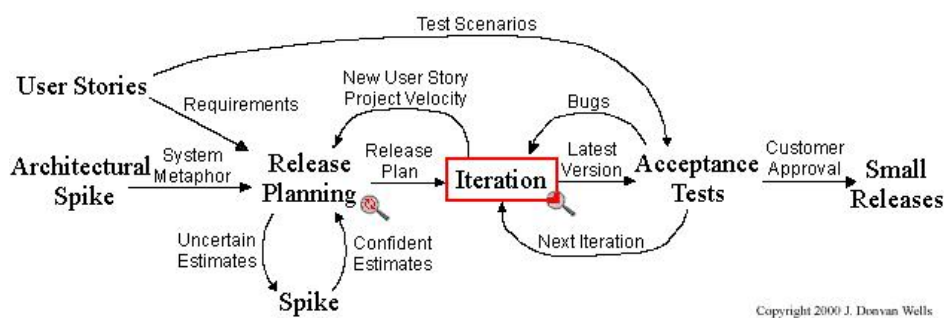


Figure 4 - Overview of XP [25]

1.11.2.2 Rational Unified Process (RUP)

Rational Unified Process (RUP) is an agile development process. It provides a structured approach to assigning tasks and responsibilities in development management. Its purpose is to ensure high-quality software that satisfies the customer within schedule and budget, as shown in Figure 5 below, the horizontal axis represents time and shows the dynamic aspects of the process. And the vertical axis shows the static aspects of the process [26].

Rational Unified Process is composed of 4 phases [22]:

Inception:

Define the system (scope) and define its relationships with the outside world, Design an appropriate system architecture, and Identify problems for system efficiency.

Elaboration:

Create an architecture to implement the functionality and impact of the architecture. Identify risks that may lead to additional costs and delays. Specify the quality requirements. Develop a project plan and estimated costs and resources.

Construction:

Capturing all the requirements and complete the analysis, design, implementation and test use cases, and managing the risks.

Transition:

- Prepare the deployment environment and create documentation for (user manuals, maintenance manuals). Fix bugs found in beta version.

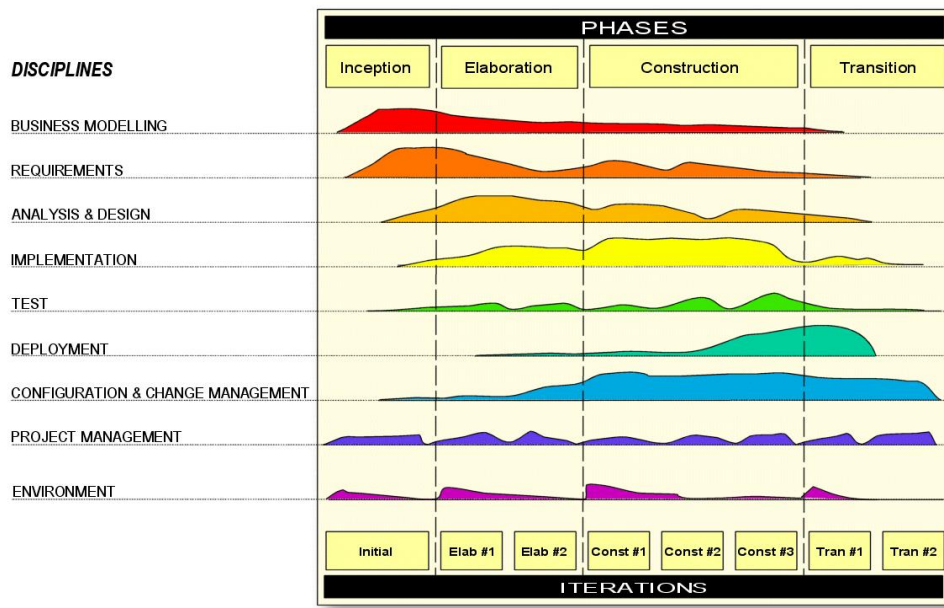


Figure 5 - A hump chart that illustrates RUP architecture [26]

1.11.2.3 Scrum

SCRUM is a framework for managing complex projects using agile principles. It is simple and easy to understand. SCRUM is based on three pillars: clarity, control and adaptation. Scrum teams deliver software iteratively and incrementally for getting feedbacks. The successive releases of the "Finished" product ensure that a viable version of the working product is always available. SCRUM rules connect events, activities and objects. SCRUM develops the product on the fly. SCRUM meetings associated with a sprint begin with a SCRUM planning meeting that defines the plan for the upcoming sprint by selecting features that can be implemented or completed during the sprint. During the sprint, there should be a daily meeting called Daily SCRUM. This is a follow-up meeting so the team can monitor and understand the situation. This meeting will discuss the work done since the previous meeting, the plan for today and the obstacles. After the sprint is completed, a sprint review meeting is held. The purpose of this meeting is to review what did and did not work during the sprint. SCRUM defines three main roles: the SCRUM team, the SCRUM owner and the product owner. The SCRUM team is a multifunctional, self-managing team; no title and all team members are responsible for the result. SCRUM master is a facilitator who helps SCRUM team, product owner and management to apply SCRUM; he is not a project manager. The Product Owner is responsible for the requirements and is the representative of the customer. SCRUM defines two items: the product backlog and the sprint backlog. The product backlog is a list of orders that contains all the requirements that have been defined, and the sprint backlog is a list that contains the requirements that will be implemented over time [27].

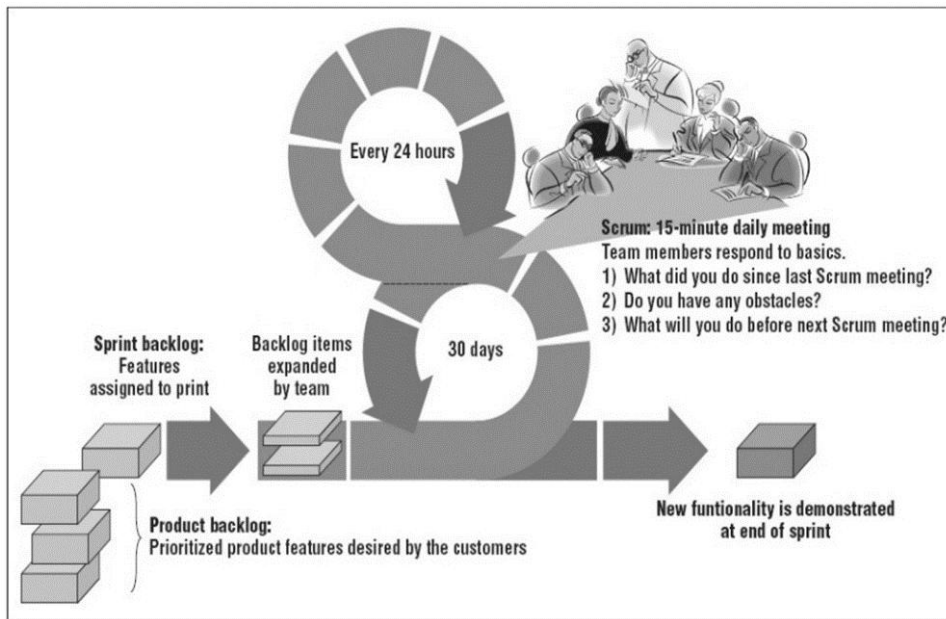


Figure 6 - Scrum overview for Agile Software Development [28]

Scrum Team

The Scrum Team consists of the Product Owner, the Development Team and the Scrum Master. Scrum teams are self-organizing and cross functional. Self-directed teams choose the best way to accomplish their tasks, rather than letting people outside the team guide them.

The Product Owner

The Product Owner is responsible for maximizing the product value resulting from the development team's efforts. How this is done can vary widely between organizations, Scrum teams and individuals. The product owner can do the above or ask the development team to do it. However, the product owner is responsible. He can express the wishes of the committee in the product backlog, but those who want to change the priority of a product backlog item must communicate with the product owner [28].

The Scrum Master

The Scrum Master is responsible for promoting and supporting Scrum. Scrum Masters achieve this by helping everyone understand the concept, principles, rules and values of Scrum. The Scrum Master is the servant leader of the Scrum team [28].

The Development Team

The development team consists of experts who work to deliver a "Done" product at the end of each sprint. A "Done" increment is required during an ongoing review. Only members of the development team perform the increment. The organization created development teams and entrusted them with organizing and managing their own work. The resulting synergy optimizes the effectiveness and efficiency of the development team [28].

1.11.3 Advantages of Agile Software Development

The advantages of agile methodology in software development are mentioned on detail in [29]:

- 1** Adapting to a changing environment: The iteration is characterized by analysis, design, and implementation and testing. After each iteration, a sub-project is sent to the client for use and feedback. The customer will approve any changes that update the software at the development stage.
- 2** Customer satisfaction is guaranteed: This process requires customers to be involved throughout the development process. The output developed after each change is sent to the user for testing and improvement only based on his feedback. The result is a high-quality product.
- 3** Documentation: Agile methodology documentation is short. The main things that should be included in the documents are a list of product features, the duration of each cycle and the date.
- 4** The risk of failure is reduced: When upgraded software is released to customers after each short development cycle and customer feedback is taken into account, developers are informed of problems that occur more than in later stages of development. It also helps to detect errors early and correct them immediately.

1.11.4 Disadvantages of Agile Software Development

The implementation of agile methodologies in project management carries certain disadvantages that demands careful consideration [28]:

- Interaction with customers is a key factor in successful software development. Therefore, if the customers do not have a clear idea about the product features, the development process will go wrong.

- **Lack of documentation:** If there is little documentation, development time is saved in favor of the agile method, although it is a big problem for developers. Here, the interior design is constantly changing according to the needs of the users after each iteration. It is therefore not possible to keep detailed documentation of the design and implementation due to project delays. Therefore, with less information available, it is very difficult for new developers who join the development team in the future to understand the software development process.
- **Saving time and wasting resources due to constantly changing requirements:** If customers are not satisfied with the software part created by an iteration and change their requirements, this additional part will not be useful. Therefore, it is a waste of time, effort and resources needed to develop that growth.
- **Better for developer management:** The agile method helps organizations make decisions about software development. However, it is very difficult for senior developers to cope with an environment that is constantly changing and every time it changes, the design and code are based only on the requirements of the time.

1.11.5 Comparing agile and traditional methodologies

One major difference between agile development and conventional development methods is that the former methodology possesses the ability to successfully deliver results quickly and inexpensively on complex projects with ill-defined requirements. Agile methods emphasize teams, working software, customer collaboration, and responding to change, while conventional methods stress contracts, plans, processes, documents, and tools [18].

Most system developers use traditional development or agile development when approaching their SDLC. Next, we will analyze the comparison between agile and traditional methods [20]:

Key Objectives: Predictability, repeatability, and optimization are key objectives for other planning-related processes. Agility quickly creates and adapts to changes.

Customer Relationships: Agile approaches work best when customers work closely with the development team and when their knowledge of the overall project is relevant. This approach carries the risk of gaps in tacit knowledge, which are mitigated by plan-based approaches that use documents, architectural review boards, and project-specific expert opinions to fill customer gaps on site.

Communication: While traditional ways favor clear and written communication, agile encourages face-to-face communication.

Requirements: Traditional methods tend to follow different requirements, but on the other hand, modern methods can maintain different requirements.

Development: Agility increases the amount of work not being done by determining computer performance based on deep documents. Although waterfall modeling relies heavily on software architecture because it is a direct function of the development process.

Table 2 summarize the differences between Agile and Traditional methodology in different aspects:

	AGILE	TRADITIONAL
USER REQUIREMENT	Iterative acquisition	Detailed user requirements are well-defined before coding/implementation
REWORK COST	Low	High
DEVELOPMENT DIRECTION	Readily changeable	Fixed
TESTING	On every iteration	After coding phase completed
CUSTOMER INVOLVEMENT	High	Low
EXTRA QUALITY REQUIRED FOR DEVELOPERS	Interpersonal skills & basic business	Nothing in particular

	knowledge
SUITABLE PROJECT SCALE	Low to medium-scaled Large-scaled

Table 2 - Agile VS Traditional [18]

Conclusion

In this chapter, we have covered the key aspects of Artificial Intelligence applications and the different development methodologies with a particular focus on agile methodologies. We explored the various types of intelligent systems, in addition to their unique characteristics and specifications. We then examined both traditional and agile development methodologies, highlighting their advantages, disadvantages, and relevance in the context of intelligent system development.

The field of software development has experienced tremendous growth and progress over the past three decades. The increasing dependence on software in various domains has necessitated the development of numerous methods and models to guide the software development process. The agile model has emerged as a popular and effective approach in software development due to its emphasis on flexibility, collaboration, and iterative development.

However, with the new type of software powered by AI, which is known for its unique specifications, there is a necessity for new development methodologies that can accommodate this new age. In the next chapter, we will delve deeper into the related work and existing methodologies and frameworks specifically designed for AI development.

Chapter 2

AI Systems Development

Methodologies: State-of-the-Art

Introduction

In the new age of artificial intelligence and the continuous growth of different AI systems, creating these AI systems is a complex and difficult task due to their nature as data-driven systems. They are very sensitive to data changes and unpredictable, which makes it obvious that traditional development methodologies will not suit these systems, but an agile approach is more suitable. However, will simply applying the agile principles solve the problem, or is there a need to adjust software engineering and development methodologies to become more effective and suitable for handling the complexities of crafting intelligent systems? In this chapter, we will discuss the related works. Our objective is to explore any existing approaches tailored to AI development. By analyzing them, we aim to answer the following questions:

- What methodologies are commonly used in the development of artificial intelligence systems?
- What are the typical life cycle stages involved in the development of intelligent systems?
- What are considered the best practices or recommended techniques for developing intelligent systems?
- What are some of the key challenges or obstacles encountered in the process of developing intelligent systems?
- What are the essential specifications or requirements that guide the development of intelligent system projects?

After collecting 13 relevant papers, we proceeded to classify them into three categories based on their type. Hence, we have established the following categories:

Case study papers: represent every paper that contains in-depth, detailed analysis of a particular instance or event, focuses on a single subject or problem, and finally summarizes the results of that study.

Review and survey papers: represent every paper that provides a comprehensive summary and evaluation of existing works on a particular topic; they usually include collecting, classifying, and analyzing the different studies paper works with the aim of identifying research gaps, trends, and areas for future research.

Original papers: represent every paper that proposes new findings, data, and concepts through research; they contribute to existing research gaps and problems by reporting the results of novel experiments, surveys, or analyses.

Table 3 shows the classification of the different articles studied in the three categories mentioned above.

Paper type	Authors	Title of the paper	Year
Case study papers	S. Dasgupta and V. K. Vankayala [30]	Developing Real Time Business Intelligence Systems the Agile Way.	2007
	M. Haakman, L. Cruz, H. Huijgens, and A. Van Deursen [31]	AI lifecycle models need to be revised: An exploratory study in Fintech.	2021
	M. S. Rahman et al. [32]	Machine learning application development: practitioners.	2023
	A. Messina and I. Voloshanovskiy [33]	Hybrid Agile Software Development for Smart Farming Application.	2020
	S. Das et al. [34]	Agile Systems Engineering in Building Complex AI Systems.	2021
	A. Vresk, I. Pihir, and M. T. Furjan [35]	Agile vs. Traditional Methods for Managing IT Projects - A Case Study.	2020
Review and survey	I. W. Syahputri, R. Ferdiana, and S. S.	Does System Based on Artificial Intelligence Need Software Engineering Method?	2020

papers	Kusumawardani [36]	Systematic Review.	
	S. Martínez-Fernández et al. [5]	Software Engineering for AI-Based Systems: A Survey.	2022
	A. El Mehdi, C. M. Yassin, and E. K. K. Eddine [22]	Survey and Comparative Study on Agile Methods in Software Engineering.	2017
Original papers	A. Abdelghany, N. Darwish, and H. Hefni [37]	An Agile Methodology for Ontology Development.	2019
	P. Kurrek et al. [38]	Q-Model: An Artificial Intelligence Based Methodology for the Development of Autonomous Robots.	2020
	M. Lourens et al. [39]	Agile Technology and Artificial Intelligent Systems in Business Development.	2022
	P. Lopes De Souza et al. [40]	ScrumOntoBDD: Agile software development based on scrum, ontologies and behaviour-driven developmen	2021

Table 3 - Paper classification

2.1 Related works

Having reviewed and analyzed the 13 selected papers, we present a summary of each paper in this section. We provide a concise overview of the key findings, methodologies, and contributions of each paper, offering valuable insights into the existing research landscape.

2.1.1 Review and survey papers

Syahputri et al. [36] conducted a systematic review of 30 papers to determine whether artificial intelligence-based systems require their own software engineering (SE) methods and standards.

The criteria used to collect the related papers are as follows: the title must contain specific sentences such as “development process”, “software engineering for AI”, “recommendation system”, and “expert system”; the content of the paper must be about developing recommendation systems, expert systems, or decision support systems; it must be about software engineering methods for AI; it must be written in English; and it must be published between 2007 and 2020.

The authors classified the papers selected in a table that includes the type of AI system, the technology of AI used, and the SE methodology used.

Finally, they proved that 75.9% of AI-based systems used SE methods as shown in Table 4, whether it was the full application of the method or just some phases, of which 61% used agile methodologies rather than traditional methodologies.

Type of AI	Software Engineering Method		Does not use SE Method
	Agile	Waterfall	
	Some/combination phase	Full	
Expert system	53.8%	7.6%	38.6%

Recommendation system	55.5%	11.1%	33.4%
Decision support system	75%	25%	0%

Table 4 - Paper's results [36]

SILVERIO MARTÍNEZ-FERNÁNDEZ et al. [5] conducted a wide review to collect and analyze the state of the art about SE for AI-based systems. The authors extracted the different characteristics of AI-based systems, identified the different existing SE approaches for AI-based systems, and identified the different challenges of SE approaches when applied to AI-based systems. They reviewed a total of 248 papers.

Some of the criteria that were used to select the relevant papers are the following: The work has been published between 2010 and 2020, it is mainly focused on SE for AI-based systems, it is not an exact duplicate of another study, and it is not a short paper of two pages or less.

They also classified the papers in a number of ways, based on the type of the paper, based on the year of publication, based on the domain, and based on SE areas.

They extracted a number of challenges in most of the SE areas, starting with the requirement phase, where they highlight the difficulties in the functional and non-functional requirements (NFR), such as the need for some new NFR types to be considered or the need to negotiate upon unfeasible 100% accuracy demands issued by customers. In the implementation phase, companies struggle to incorporate the different tools of AI and ML. In testing phase, there is a lack of automatic solutions as well as the difficulty of collecting enough testing data. In model building, data poses several challenges, such as ensuring high-quality data and enough varieties, there is a lack of standards methods for preparing, training, and validating datasets as well as defining standards quality measures for example ensuring that the model won't reinforce existing discriminations (on gender, race, or religion). The rest of the challenges are related to managements, infrastructure and maintenance.

2.1.2 Case study papers

Dasgupta and Vankayala [30] talk about the competitive nature of business and the contentiously changing market; thus, a flexible development process is required to accommodate the rapidly changing requirements and market expectations. The iterative and incremental product development principles in agile methods can guarantee the right flexibility.

The paper is supplemented with a case study for developing a real-time business intelligent system. The methodology used for the development process is scrum with a test-driven approach. It consists of planning the different tests starting from the requirement phase.

They followed the scrum phases, delivering incremental builds and continuous integration every sprint while making sure the client was always available during the development process.

They extracted a number of benefits from applying the scrum methodology, including that test-driven development ensures a clean release, progress is always guaranteed, even if requirements are not clear since the focus is on high-priority requirements until other requirements are communicated, and dividing the project into multiple patch sets makes the project more achievable.

Haakman et al. [31] conducted a case study at a global bank known as the ING Bank on the topic of AI life-cycle models. They selected and interviewed 17 practitioners with different roles in the process of developing machine learning applications from different departments of the enterprise. The authors focused on the machine learning life cycle and reported the three most common life cycles that exist.

The Cross-Industry Standard Process for Data Mining (CRISP-DM) shown in Figure 7. It consists of six activities, as follows: It typically starts with business understanding to determine business objectives, then data understanding, followed by data preparation to make data ready for modeling. The produced model goes through evaluation tests for deployment.

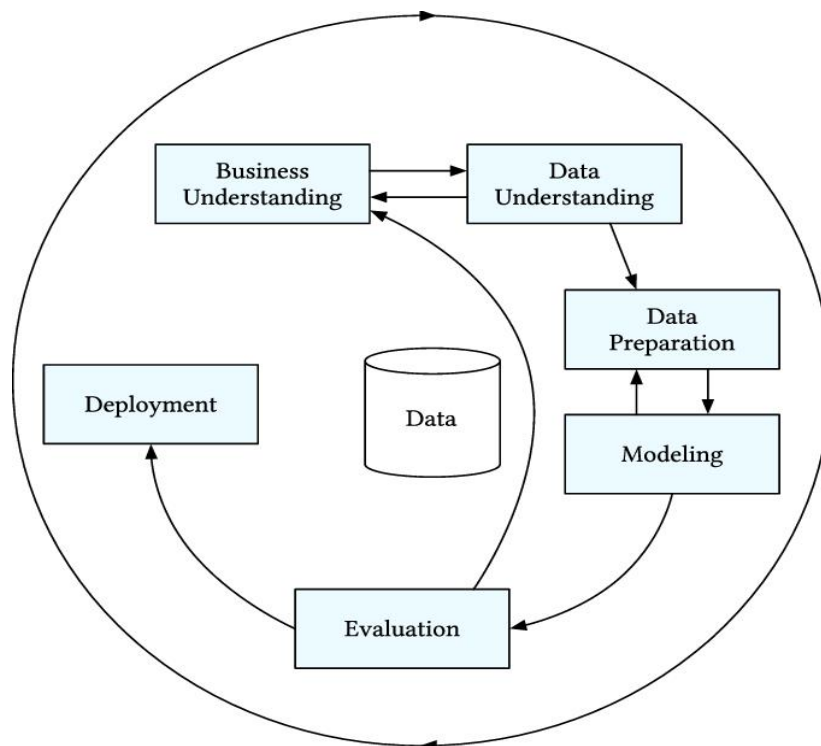


Figure 7 - The Cross-Industry Standard Process for Data Mining (CRISP-DM) [31].

The Team Data Science Process (TDSP) is an agile, iterative data science methodology, proposed by Microsoft to deliver machine learning solutions efficiently. It is composed of four main phases as shown in Figure 8: business understanding, data understanding, data acquisition, modeling, and deployment.

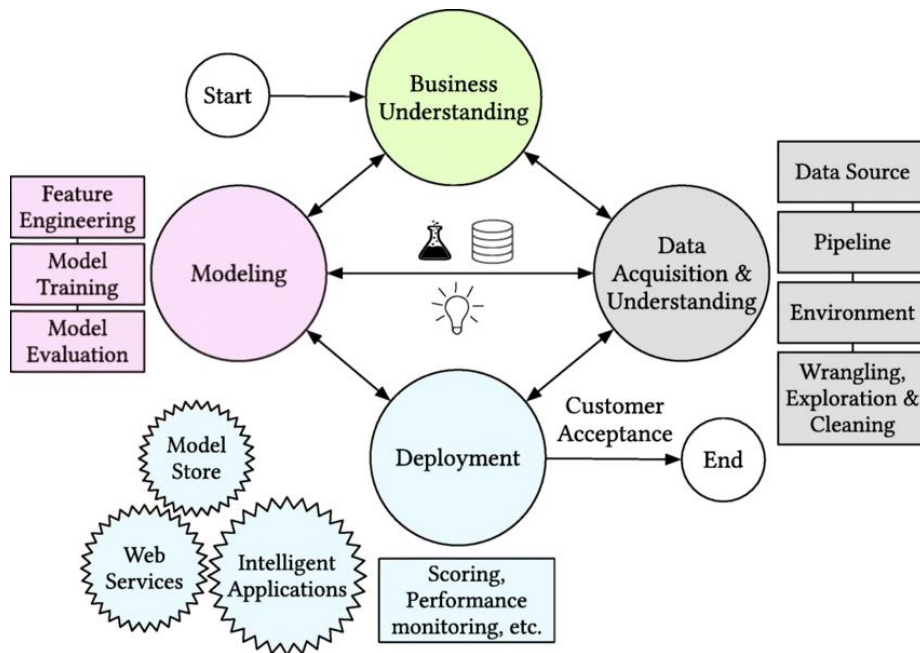
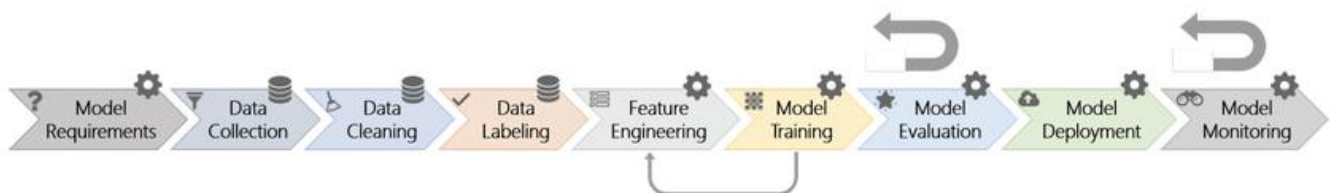


Figure 8 - The Team Data Science Process TDSP [31].

The Microsoft model described by Amershi et al. [41] includes nine steps (see Figure 9): model requirements, data collection, data cleaning, data labeling, feature



engineering, model training, model evaluation, model deployment, and model monitoring.

Figure 9 - The Microsoft model described by Amershi et al [31].

Later on, the authors proposed a refined version of these 3 life cycles, we mention the CRISP-DM life cycle where they added the following activities, as shown in Figure 10: data collection, a go-no-go checkpoint for feasibility assessment, documentation, risk assessment, and model monitoring.

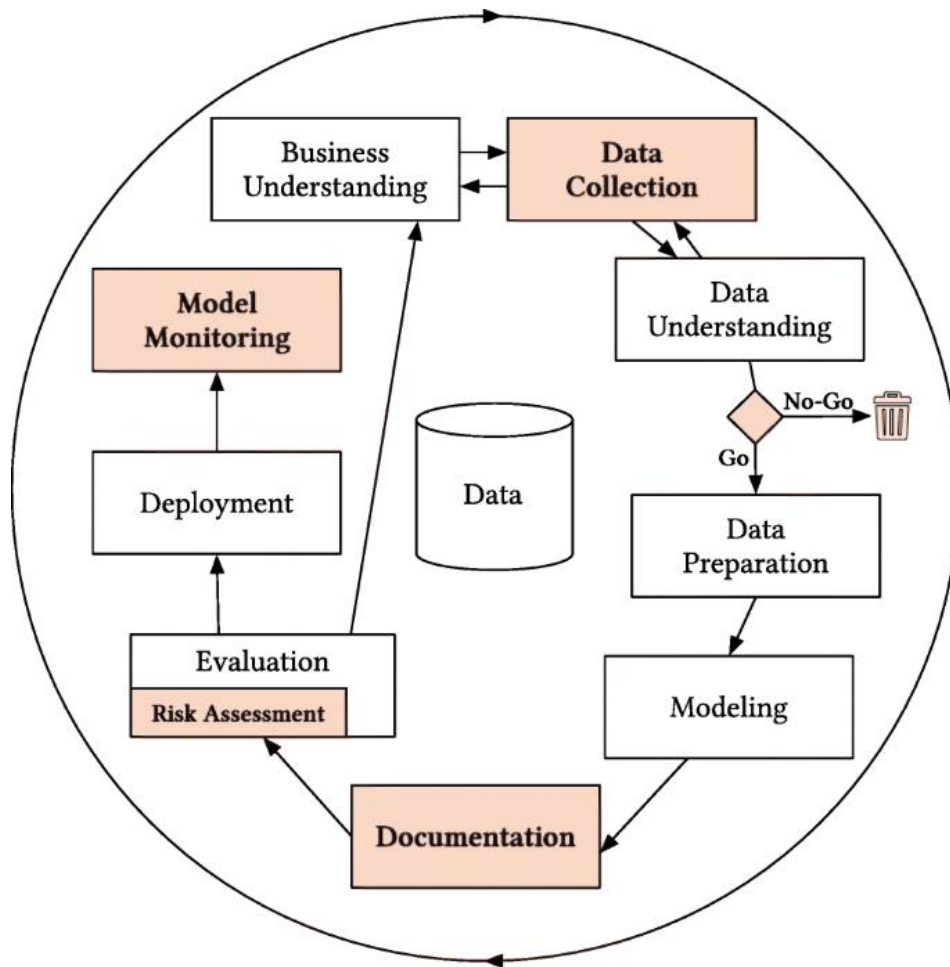


Figure 10 - Refined CRISP-DM model. Additions in red, with bold text [31].

After observing the interview results, they extracted a number of findings and challenges as follows: A feasibility study at the first iteration is crucial for the enterprise to determine whether it is worth it to continue the development (fail-fast approach). Collecting, understanding, and preparing data are the most time-consuming stages of machine learning projects. Documenting and keeping track of experiments is essential. The paper also highlighted the need for more automation for testing, monitoring, and governance of models. The challenges extracted include model governance and risk assessment, auditing of AI technologies before use, and a lack of clear performance metrics standards.

Another case study was published by Rahman et al. [32]. They selected 80 practitioners with diverse skills and experiences in developing machine learning applications to gather insights about trends, challenges, and best practices for the development process of ML applications.

The selection of practitioners was based on their self-declared profiles in the professional network LinkedIn, as well as their contributions to the development of ML applications based on their GitHub repositories. While making sure that they were attached to ML/AI application development, they focused on four main phases of the ML life cycle: data collection and processing, feature engineering, model building and testing, deployment, and monitoring.

The authors summarized 17 findings. The key takeaways are as follows:

- Practitioners mostly use agile methodologies for the development of ML applications.
- The quality of the input data is very important for the performance of the model. Feature engineering is a very important task for training a good model. Practitioners have declared the use of multiple techniques, such as statistical analysis and domain expertise. In general, data processing tasks are crucial steps in the development of ML applications, and it can be a time-consuming process.
- Some of the challenges reported by practitioners are the lack of high-quality data. Automating data cleaning is another reported challenge. There is a lack of standards; for example, there is no standard definition of “clean data”. Another issue that has been emphasized by practitioners is the need for domain expertise.

Practitioners also pointed out a few challenges in model deployment, including difficulties in obtaining real data to test the model in production, difficulties in maintaining the correct form and type of data coming to the model, the infrastructure where the model will operate, and the need to evaluate the model with different business metrics.

The authors observed a number of best practices for evaluation techniques that include: performance measuring based on other famous datasets, performance evaluation compared to a baseline model, performance evaluation compared to other models with different languages, and performance evaluation with domain experts.

Messina and Voloshanovskiy [33], in the context of the final project of the Master of Science in Information Technology-Software Engineering, they conducted a case study about building a smart farming application in agile way and adopted the SCRUM methodology. The authors reported a knowledge gap issue, causing uncertainty and hesitation about which functionalities should be included in the application.

They added a new phase at sprint 0 (the inception phase) as well as enlarging the role of scrum product owner (PO) to cover the knowledge gap. Besides that, they decided to engage a domain expert in the development process.

The goal of the inception phase is to help understand the application domain, investigate the domain, and come up with ideas for the basic functionality needed for the final product. The PO is responsible for selecting and analyzing the articles and different internet resources concerning the application domain, communicating and interacting with experts in the domain (in their case, real farmers), and a number of other roles concerning the business side, such as investigating competitors. Finally, the PO communicates his findings with the development team.

In conclusion, they highlighted the term "hybrid agile", agile principles cannot avoid the need to include hybrid solutions, something different from the usual software engineering activities.

The paper by Das et al. [34] argues that due to the nature of data being noisy, inaccurate, and incomplete, the outcome of AI systems can often be vague and unclear. Thus, an AI project has to be agile in nature; in other words, it has to follow an iterative mindset.

To confirm that the authors conducted a case study at AdventHealth to build a decision-support intelligent system, they followed the scrum framework and used various AI technologies, including natural language processing, machine learning, and deep learning.

Their experience shows that an agile mindset is highly demanded in the development of AI systems due to the questions asked during the development process that lack definitive answers, such as how much data we need, what algorithms to use, what labels we need, or what level of accuracy is achievable. The authors also confirm that agility principles with the Extract Transform Load process contribute to a faster and easier workflow to iterate on data.

In general, the agile approach provides a quick turnaround time for incremental delivery and meeting client requirements.

A survey of 16 IT experts was conducted by Vresk et al. [35], with the title Agile vs. Traditional Methods for Managing IT Projects. The authors selected experts from a company that switched to using project management according to the agile approach as opposed to the previously used traditional approach. The experts have experience with both approaches, which makes them ideal to study and compare both approaches. In their paper, they analyzed a number of methods, such as waterfall, PRINCE, PMBOK, and SCRUM. They explored the different methods in many ways and compared them with different criteria, which include:

- User requirements in traditional methods (TDM) are clearly defined at the beginning, whereas in agile methodologies (AGM), the requirements aren't clear and vague.
- Team size in TDM usually requires larger teams, while AGM can function with smaller teams.
- The client in TDM is absent most of the time, in contrast to AGM, which demands constant client feedback.
- In TDM, it is very hard to make changes after the development starts; however, in AGM, it is highly flexible and accepts changes.

Finally, they concluded a number of advantages and disadvantages of both agile and traditional methods. The key points include that in TDM, there is less time lost at meetings, the requirements are clear and defined, and problems are predictable, but it lacks flexibility to adapt to changes, and it is slow with rare deliveries, whereas in AGM, it is highly flexible with good team cooperation and frequent deliveries. Even though it is difficult to hold to deadlines in agile approaches due to the frequent changes as well as the time lost in the daily meetings, it is agreed that agile methods are a better choice for most IT projects.

2.1.3 Original papers

Abdelghany et al. [37] propose a methodology based on agile principles for developing ontologies. Although ontologies might not be an example of fully AI systems, they share common characteristics. Ontologies and AI systems both require domain understanding and knowledge capture; these two activities play a crucial role in the development of these systems. With that in mind, the authors propose a methodology based on scrum methodology for developing ontologies (AMOD), which includes three main phases: pre-game, development, and post-game (see Figure 11). AMOD also includes the following roles: ontology owner, ontology engineer, and ontology user. They propose two main activities within the AMOD methodology that could serve us in AI development: knowledge acquisition for collecting the different data and conceptualization to organize the knowledge into semi-formal representation. Finally, they tested their methodology with a case study, and it performed well according to IEEE standards compared to other existing methodologies.

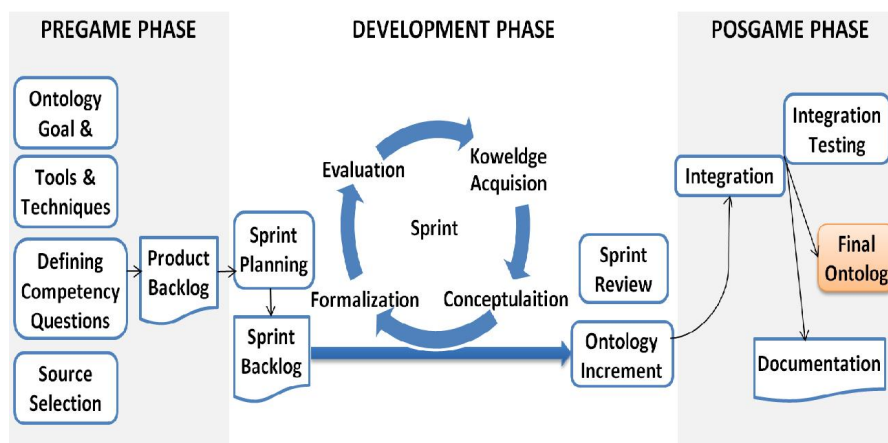


Figure 11 - AMOD workflow [37].

Kurrek et al. [38] provide an overview of existing development methodologies for the development of autonomous robots. While focusing on AI-based methodologies, they propose a new framework named the Q-Model. It integrates AI technologies, mainly reinforcement learning, for creating the behaviors of the robot by self-learning instead of the traditional way of hard coding the controls and manually testing and validating them.

The Q-model has four main phases presented in Figure 12: system design, module control, module verification, and system application. AI forms the central point of these four steps and serves them all.

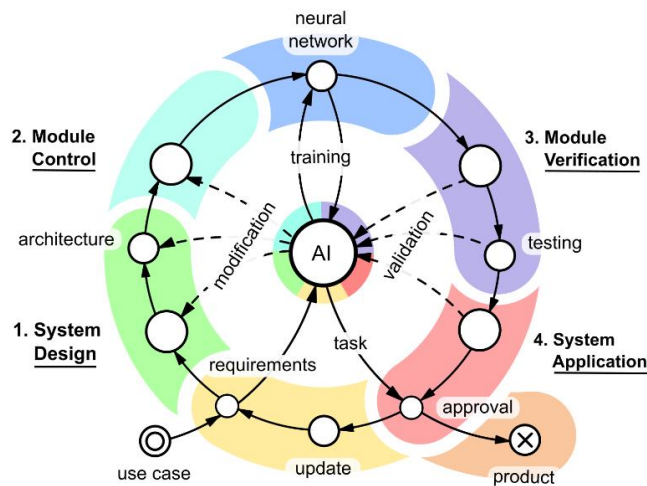


Figure 12 - Q-Model workflow [38].

The authors describe different stages from non-AI-based development to fully AI-based methodology. The Q-model aims to make the development process fully automated by training a neural network model. The design, control, verification, application, and update processes are done by the AI agent. While the proposed approach can save a significant amount of time and effort for developers, it is still out of reach at the current time. It is not clear how AI is actually implemented on the technical side.

Even though the authors provided a case study where they implemented the Q-model for building a smart robot agent in a simulation environment run by the Unity Game Engine, it is not reliable because, in reality, developing such a robot can be quite challenging and more difficult. Also, relying on AI as a self-learning tool is not 100% reliable since AI models can be unpredictable.

Lourens et al. [39] explain how business owners are turning to artificial intelligence for what it provides for the company in terms of various benefits and value. The authors use as an example the supply chains of Moroccan logistics firms. They list a number of benefits and functions of implementing AI in the supply chain, including identifying anomalies and faults, forecasting inventory, demand, and supply, and enabling firms to eliminate non-value-adding operations and concentrate on the most productive ones.

The authors report the use of agile development methodologies for software development in general and for intelligent systems in particular (see Figure 13). They highlight the property of continuous testing and integration for AI and ML systems, which is in compliance with agile principles.

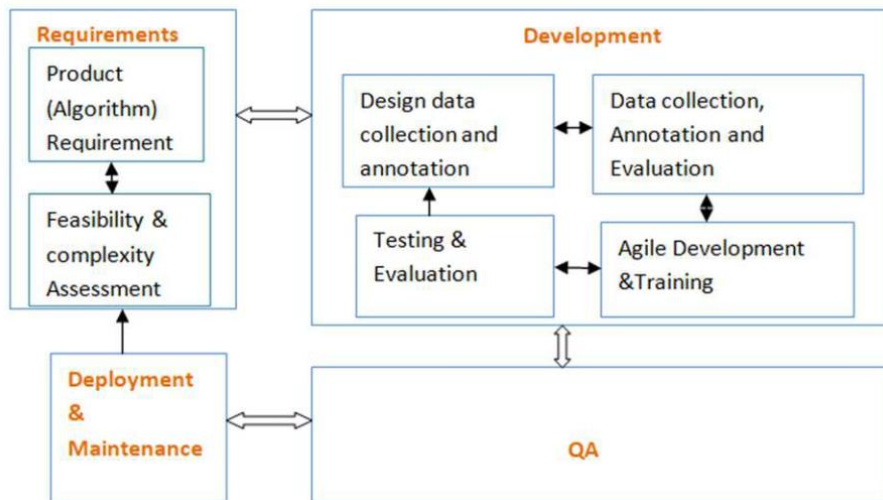


Figure 13 - Agile workflow for AI applications [39].

The paper suggest incorporating AI into the development process. Examples of incorporating AI techniques are coding helpers, an intelligent coding assistant trained with machine learning that can check for bugs and suggest code snippets; an accurate estimator for development time and cost; and AI for project planning and management.

Lopes De Souza et al. [40] address an issue that frequently occurs in every development project: having to completely redefine some system behavior scenarios due to ambiguities in the requirement specifications caused by using neutral language. The authors consider the use of ontologies and behavior-driven development (BDD) with the Scrum framework and propose a new methodology called ScrumOntoBDD.

The proposed approach starts with two main phases that can be performed in parallel as shown in Figure 14:

- Creating Domain Application Ontologies: This phase is responsible for creating an ontology that covers domain ambiguities, takes as input domain documents and user stories in natural language, and has an output of ubiquitous language terminology.
- Extracting Application Requirements: This phase has an input of user stories and different behaviors of the system and an output of formally represented requirements.
- The rest of the phases follow the scrum framework: building product backlog, defining sprint backlog, executing sprint, sprint review, and retrospective meeting.

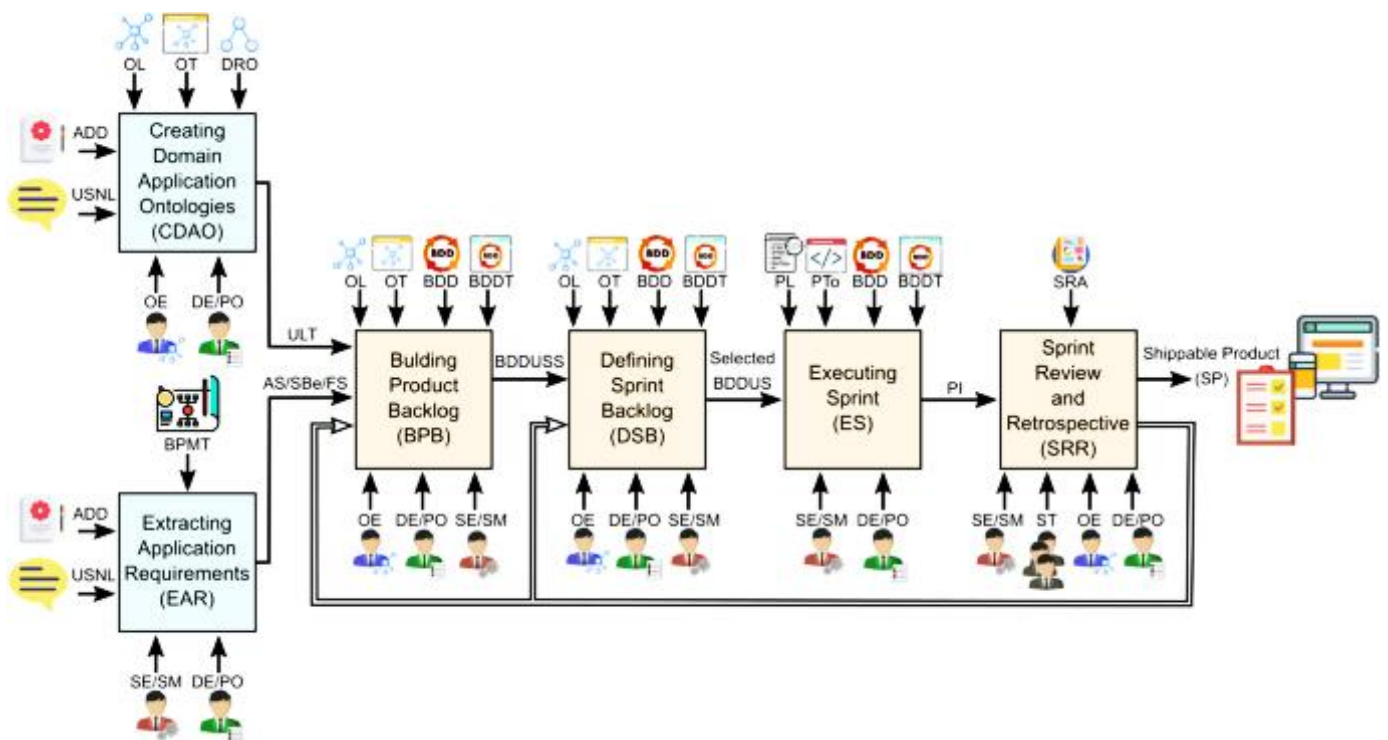


Figure 14 - ScrumOntoBDD workflow [40].

The paper is supplemented with a case study, applying the ScrumOntoBDD approach to build a software system in the field of education. They verified that the employment of ontologies resolved the semantic problems and reduced the ambiguities initially with the project's requirements in the product backlog. Integrating the BDD allowed the project owner to better follow up and communicate with developers throughout the development process. The main challenge of this approach is the number of new roles needed (domain expert, ontology engineer, software engineer) compared to the traditional scrum. This makes the ScrumOntoBDD approach costly and not suitable for a limited number of participants.

2.2 Comparative table

In the following table, we will mention each methodology used in the papers analyzed before with their own advantages and disadvantages

Article Title	Approach Name	Advantages of the Approach	Disadvantages of the Approach
ScrumOntoBDD: Agile software development based on scrum, ontologies and behaviour-driven	ScrumOntoBDD	Facilitated, enhanced project owner involvement and improved communication with developers throughout the development process.	Number of new roles needed (domain expert, ontology engineer, software engineer) compared to the traditional scrum.
Agile Technology and Artificial Intelligent Systems in Business Development	Agile development with incorporating AI tools	Automatic testing, and coding assistant.	Difficulties in selecting the right AI tool and integrating them.

<p>Q-Model: An Artificial Intelligence Based Methodology for the Development of Autonomous Robots</p>	<p>Q-Model</p>	<p>Reduce the time and effort expended by developers.</p>	<p>Relying on AI as a self-developing tool is not 100% reliable.</p>
<p>An Agile Methodology for Ontology Development.</p>	<p>agile methodology for ontology development (AMOD)</p>	<p>They propose two main activities that could serve us in AI development: knowledge and conceptualization.</p>	<p>Ontologies might not be an example of fully AI systems.</p>
<p>Developing Real Time Business Intelligence Systems the Agile Way.</p>	<p>Scrum with test-driven development</p>	<p>incremental builds and continuous integration, clean release, progress is always guaranteed.</p>	<p>Use of regular Scrum with no adaptation for AI systems, (lack of data related phases like data acquisition and data processing).</p>
<p>Agile Systems Engineering in Building Complex AI Systems.</p>	<p>Scrum</p>	<p>faster and easier workflow to iterate on data.</p>	<p>Use of regular Scrum with no adaptation for AI systems.</p>
<p>Hybrid Agile Software Development for Smart Farming Application.</p>	<p>Scrum with new phase and roles</p>	<p>Better understanding of the application domain.</p>	<p>The product owner now require much more experience in data acquisition.</p>

Table 5 - Comparative table

2.3 Summary

In summary, even though many studies have been published about AI system development methodologies, there is still a lack of standards. The papers we have seen fail to agree on common standards to use in the development of these systems. Each paper treats the problem with different techniques, models, or roles. Nevertheless, they noted that developing AI systems is different from developing regular software. AI system operation is based on data and knowledge, and not a hard-coded program; it requires the collaboration of data scientists and software engineers, which creates conflicts and other problems. From this, literature review, we have extracted a number of problems, as follows:

Data-related problems represent the majority:

- Collecting sensitive data requires different access levels and privileges.
- Dealing with sensitive data limits the tools and platforms available for data scientists.
- All project dependencies need to be previously approved and audited by an enterprise. This can be frustrating because practitioners have to wait before they can explore the potential of new technologies.
- Understanding the domain and business is an important factor for collecting the right data, processing the data, and evaluating the model; thus, the integration of domain experts is crucial.
- In general, handling the data in AI systems is much more complex than handling traditional data in databases. As a result, it is a long process and requires an agile process.
- Lack of standards: for example, there is no clear definition of clean data, and there are no standards for what accuracy is considered good enough.

Agility-related findings include:

- Being agile is not straightforward in the early phases of machine learning projects. They argued that understanding the business, collecting data, and treating it in the first phases do not fit in a small iteration.
- The development process is not standardized, it is sometimes tailored to fit specific application development contexts. This can be done by adding special sprints or roles depending on the requirements of the project.

- Agility has drawbacks, such as the never-ending loop of adding new features and spending a lot of time in frequent meetings.

Documentation and automation-related problems they involve:

- Lack of documentation, model tracking, and data tracking due to frequent changes.
- Lack of automation for documentation, data processing, model training, and deployment.

Model maintenance and governance are usually neglected; there is a need for a special phase for model testing and risk assessment after the deployment.

In general, agile methodologies like Scrum and XP are widely used for AI system development due to their flexibility and iterative nature, which are better suited to the unpredictable and data-driven nature of these systems, compared to traditional waterfall and V-model approaches. The development life cycle for intelligent systems faces unique challenges across various stages, including requirements engineering, data management, model building, testing, and deployment, requiring adaptations and enhancements to existing software engineering practices.

Conclusion

The review of the related works has revealed the limitations of existing software engineering approaches in effectively addressing the unique challenges of developing artificial intelligence and intelligent systems. While agile methodologies have shown promise, there is a clear need for a more tailored methodology that can fully handle the complexities inherent in creating advanced AI-powered systems. In the next chapter, we will propose an adapted software engineering framework specifically designed to guide the development of intelligent systems, drawing insights from the key findings and gaps identified in the current literature.

Chapter 3

ScrumAI: Adapting Scrum Agile Methodology for AI Development

Introduction

The development of AI systems is different from the development of traditional software due to the challenges identified in the previous chapter, ranging from data handling to documentation, automation, and model maintenance. Thus, there is a need for a structured approach that can present solutions for the problems identified previously.

In this chapter, we propose an improved scrum methodology tailored to address the specific challenges in AI projects. Our methodology aims to standardize processes, enhance collaboration, and ensure robust model maintenance and governance. By integrating best practices from both data science and software engineering, we seek to create a flexible yet structured framework that can adapt to the dynamic nature of AI development.

We will begin by identifying the problem. Next, we will provide a detailed explanation of our proposed methodology. Finally, we will explore how our methodology effectively addresses the majority of the issues we have identified.

3.1 Problematic

Despite the successes achieved by agile methodologies such as scrum in developing various software, they are no longer sufficient for developing the new generation of software represented by artificial intelligence software or intelligent systems. This is because these methodologies are not specifically designed for developing intelligence systems.

This is confirmed by the previous chapter, as the different papers highlight the clear difference between artificial intelligence software and regular software, and therefore there are questions as to whether the existing approaches are sufficient to develop such systems or whether there is a need to upgrade these approaches or propose new ones that are compatible with the characteristics and needs of intelligent systems, which include the following:

- data acquisition and data processing,
- the integration of new roles in the development process such as domain experts,
- the changing nature of intelligent systems and continuous experiments even after production
- collaboration between data scientists and developers,

- the appearance of new non-functional requirements such as high accuracy, low error, business risk, model output safety.

This raises many questions like: Is it necessary to design a new development methodology for AI systems, or to adapt existing ones? If the latter, how do we adapt existing methodologies like scrum to fit the life cycle of the process of intelligent systems? How do we keep track of data changes and model changes? Do we need to add new, specific roles? What workflow should be followed in order to make the development easier and faster?

In the coming sections, we will explore scrum as an adapted methodology that will attempt to discuss these challenges and provide possible solutions.

3.2 Scrum-IA approach

Our proposition consists of adapting the scrum methodology for the specifications of intelligent systems and the challenges mentioned before. We added a new cycle (sprint) for the development of the machine learning models in addition to the traditional software cycle, as shown in Figure 15. Furthermore, we assigned new roles, including domain experts, cognitive scientist², and the data scientists team for data operations (data-ops team). We also added a new meeting that can be arranged at any point to exchange the necessary information between the development team (dev-ops team) and the data-ops team during the sprint.

² A cognitive scientist is someone who studies the human mind and its processes, such as perception, attention, memory, reasoning, and language.

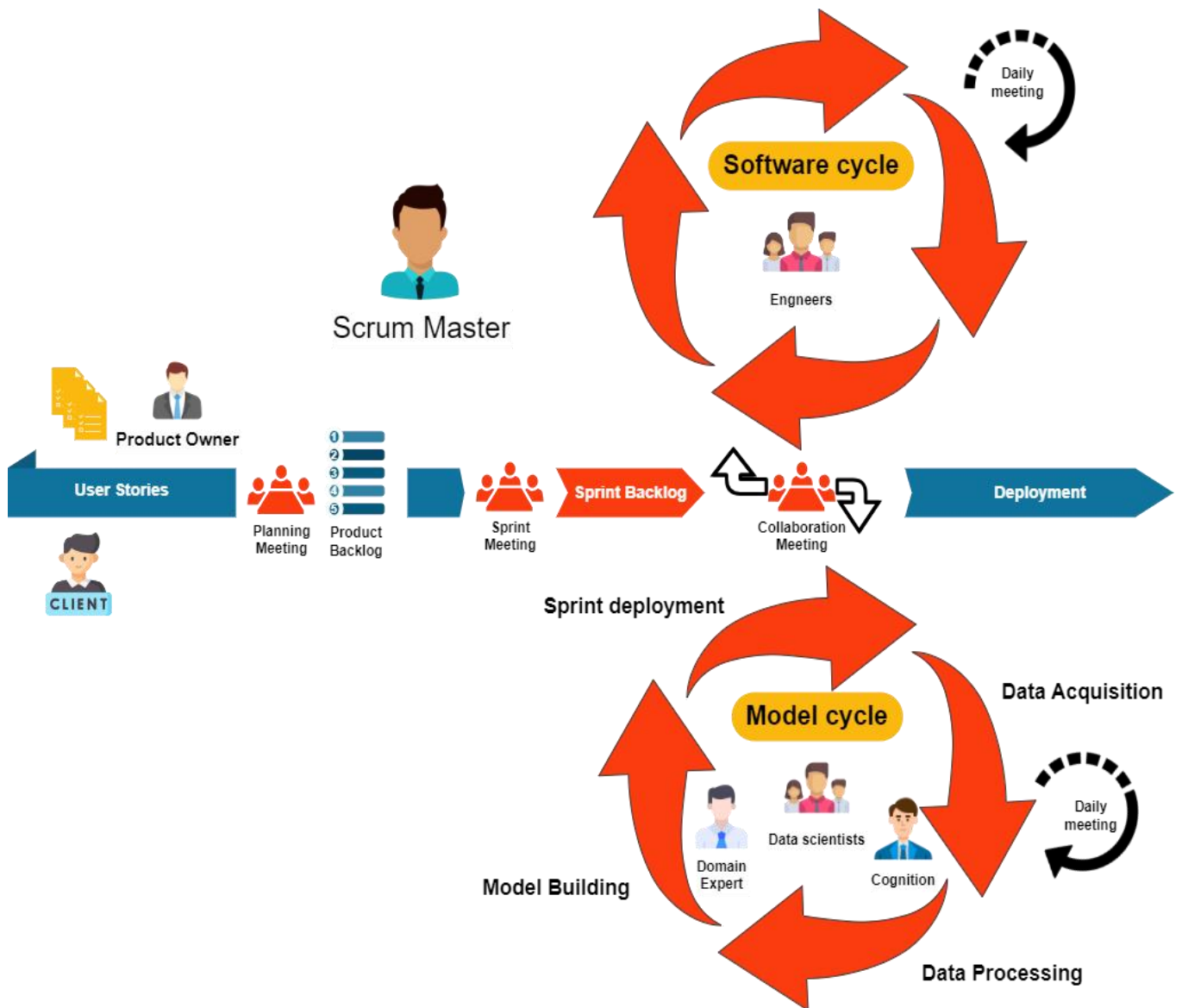


Figure 15 – The workflow of ScrumAI

Finally, we emphasize some best practices, including: collaboration between the dev-ops team and the data-ops team, integrating domain experts in the development and evaluation process, defining clear boundaries and clear model-evaluation measures for the project at the start in order to avoid the phenomenon of the never-ending loop of continuous integration, documenting briefly each sprint at the end while focusing on documenting main activities such as data processing and model building, using data version control to keep track of data and model experiments, and automating tasks such as data cleaning and processing using scripts and data pipelines.

3.2.1 Model cycle

Data Acquisition: In this phase, the Data-Ops team strives to gain a comprehensive understanding of the data domain, establish precise requirements, execute data collection activities, and evaluate its efficacy in aiding the problem-solving process.

Data Preprocessing: In this phase, the Data-Ops team conducts data cleaning activities through graphical visualizations, encompassing the identification and handling of missing values, outliers, duplicates, and data format inconsistencies present within the dataset. These issues are thoroughly examined and addressed through appropriate actions, such as modifying or eliminating them, to ensure the data is cleansed and prepared for subsequent stages of modeling.

Model building: Following the data preprocessing phase, the data-ops team identifies the key entities and their attributes, then proceeds to select the appropriate model for training the data. Additionally, if the chosen model necessitates data transformation, it is performed accordingly. Subsequently, the data is split into training and testing sets.

Model validation: Once the model is applied, a validation process is put into practice to ensure its alignment with the predefined requirements and expectations. This validation is performed by some techniques, such as cross-validation, evaluation metrics (MAE, MSE, RMSE, etc.), a confusion matrix, performance visualization, or even validation domain experts. The documentation is an obligatory component of this phase for capturing the design decisions, assumptions, and rationale underlying the chosen data model. This documentation serves as a valuable resource for comprehending the data model over time.

Sprint Deployment: At the end of each sprint, the trained model is deployed and packaged in a deployable format to be integrated with the application, either locally or online.

3.2.2 Functioning of ScrumAI

In this section, we will offer a comprehensive and detailed explanation of Scrum-AI and its main phases. This approach integrates a model life cycle within the Scrum framework alongside the traditional software life cycle. Our focus will be on the model life cycle, as it represents the new addition in the Scrum-AI methodology.

3.2.2.1 Main phases of ScrumAI

Here are the key phases of the Scrum-AI:

Sprint Zero: it represents the initial planning phase which involves two roles, the product owner and analytic manager. In this phase, the client's requirements are discussed and declared by the product owner, as well as the project's cost and the project's life. In Scrum-IA, we recommend defining clear boundaries for the project, which involves establishing the scope and clear definition for what is considered clean data and a good model, what is the acceptable error value, and what is the required accuracy, as well as defining non-functional requirements for the model, such as business safety and the model's output safety.

Clear boundaries reduce the never-ending loop of continuous integration and ensure focused and well-defined objectives for each iteration, reducing scope creep and maintaining project alignment. In addition, the technologies needed in the development process, such as libraries and tools, are discussed in this phase as well. The analytic manager will define, based on requirements and project life, not only the number of sprints needed but also the number of developers needed. The outcome of this phase is a complete product backlog with all total user stories and a project timeline to follow.

Sprint meeting: this meeting happens at the start of every sprint and involves the following roles: product owner and development team, which consist of software engineers, data scientists, and domain experts. In this meeting, they go over the product backlog and define a number of user stories to develop in the coming sprint. The outcome of this phase is a sprint backlog.

Sprint: A sprint is a short iteration that usually lasts between 2 and 4 weeks. It is a work cycle where the development team works to complete a set amount of work. The sprint in Scrum-AI is decomposed into two different work cycles that can happen in parallel or in sequence; thus, each work cycle can have its own specific development team or they can share one team depending on: what features are being developed in the sprint, the number of developers available, their competence, and their preferences since the development team in Scrum is self-organized.

The first work cycle is the software cycle, which is managed by a dev-ops team that includes software engineers. The second work cycle is the model cycle, which is managed by the data-ops team and includes data scientists, cognitive scientist, and a domain expert. As mentioned before. The model cycle includes the following main phases: data acquisition, data processing, model building, and model evaluation.

During a sprint, the teams check in during the daily standup meeting about how the work is progressing. The goal of this meeting is to discuss any blockers and difficulties that would influence the team's ability to achieve the sprint goal.

Critical aspects of ScrumAI include comprehensive documentation and the involvement of a domain expert, as well as cognitive scientist.

An expert in the application's domain must be integrated into the development process. Their functionalities range from:

- **Data understanding:** Bridging the gap between domain-related terms and data scientists.
- **Data Source Identification:** Identifying and validating relevant and high-quality data sources that can be used in model training.
- **Feature Engineering:** Providing insights into important features and suggesting domain-specific features.
- **Performance Metrics:** Choosing meaningful performance metrics and validation techniques.

Model documentation is another important aspect of Scrum-AI, which details the iteration process of model building at the end of each sprint, including data processing, model implementation, and model parameters, enabling the data-ops team to easily recall data preprocessing procedures already used in previous versions.

Finally, in ScrumAI, we recommend the use of data/model version control systems to keep track of data and model previous experiments.

The output of the sprint should be a set of ready-to-use software functionalities to be tested as well as a machine learning model with a brief documentation and a data/model version.

Collaboration meeting: this meeting is specific to ScrumAI in the case of having two different development teams for the software life cycle and the model life cycle. It can be arranged at any point during the sprint if necessary. The main goal of this meeting is to exchange the necessary information between the dev-ops team and the data-ops team in order to keep the workflow of the development process going and avoid long-term errors that can cost more time and errors. In addition, it helps emphasize team collaboration. For instance, in an intelligent system for predicting machine failure, the dev-ops team might need to know which attributes (sensors) are used in the prediction model in order to implement the right inputs that correspond to the right sensors in the user interface.

Sprint retrospective meeting: this meeting happens at the end of every sprint and serves the purpose of identifying areas for improvement for the next sprint.

The workflow of Scrum-AI is shown with more details in Figure 16.

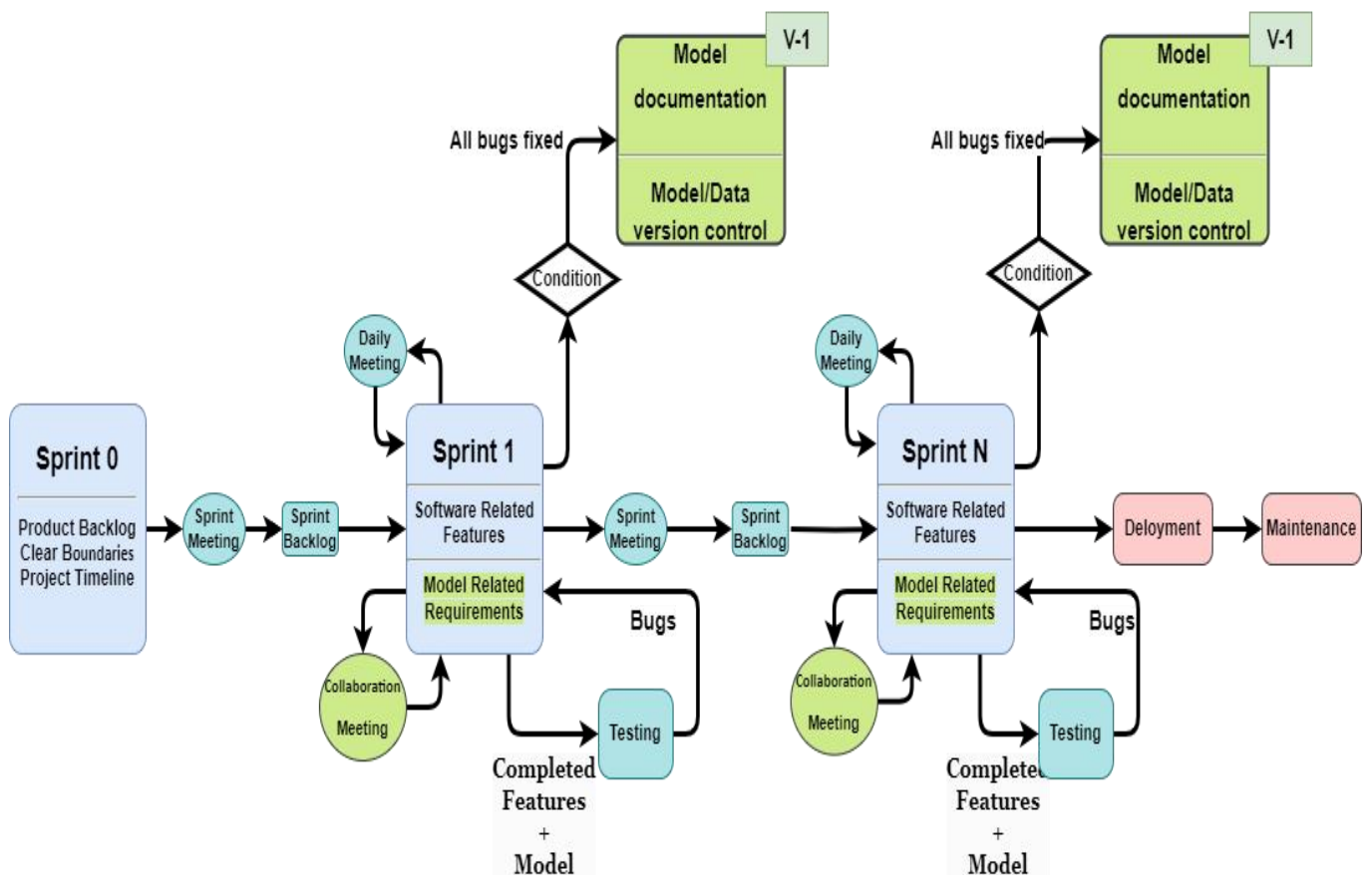


Figure 16 - Detailed workflow of ScrumAI

3.3 Addressing AI development challenges with ScrumAI

In this section, we will present evidence supporting our choices and demonstrate how our approach effectively addresses the previously identified problems. We will begin by reviewing the specifications of intelligent systems and the challenges encountered in their development, and then we will explain how each difficulty was resolved through our approach.

3.3.1 Specifications for intelligent systems

Developing intelligent systems presents several challenges due to their unique specifications, including the following: The development of intelligent systems is highly data-driven; thus, data acquisition and processing are fundamental. Intelligent systems development requires the integration of new roles like domain experts to help with domain-related data. The dynamic nature of intelligent systems necessitates continuous experimentation, even after deployment. Collaboration between data scientists and engineers is crucial. Finally, as new non-functional requirements emerge, such as high accuracy, low error rates, business risk management, and model output safety, they need to be clearly defined.

3.3.2 Addressing the challenges

Scrum is the most widely adopted agile methodology in the industry, it has clearly defined roles, and it offers agility and effective management capabilities. Additionally, Scrum is scalable and can be applied to projects of varying sizes and complexities. Despite some drawbacks, the benefits of Scrum outweigh the negatives. We believe that the capabilities of Scrum (agile, flexible, iterative, and incremental) align perfectly with the continuously changing and experimenting nature of intelligent systems.

In addition to handling the experimenting nature of intelligent systems, ScrumAI also tackles the following problems:

- We trust that data acquisition and data processing activities are well handled in ScrumAI through the separation of work cycles and the integration of the model cycle in the Scrum sprints.
- We trust that an effective collaboration between data scientists and engineers is achieved through the collaboration meeting.

- We defined some new non-functional requirements that need to be considered in every intelligent system's development such as model risk assessment, which controls and limits the output of the model, high accuracy and low error which represent how much accurate is the output of the model.
- Keeping track of the data and model development experiments is obtained by emphasizing the use of data and model version control platforms and the practice of model documentation at the end of every sprint.

Conclusion

In this chapter, we have proposed an adapted methodology called Scrum-AI, which aims to address the unique challenges faced in developing AI-based systems. By adapting the traditional Scrum framework and integrating it with a dedicated model life cycle, Scrum-AI provides a structured and collaborative approach to AI development.

The key aspects of Scrum-AI include clear project scoping, the integration of domain experts and cognitive scientist, comprehensive documentation, and the use of data and model versioning tools. These features help to overcome the issues identified in the previous chapter, such as data handling, model maintenance, and the dynamic nature of AI systems.

While the Scrum-AI methodology has been tailored specifically for machine learning-based systems, it can be adjusted slightly to fit other types of intelligent systems as well. By modifying the model life-cycle phases to other intelligent systems life cycles, the Scrum-AI approach can be adapted to a wider range of AI-driven applications for example in the case of an expert system, the model cycle can include the following phases: data acquisition, data processing, knowledge base building, inference engine building, test and deployment.

In the next chapter, we will explore a case study to validate the effectiveness of the Scrum-AI methodology. By applying this approach to a real-world intelligent system, we will demonstrate how it can make the development process easier and ensure a successful intelligent system.

Chapter 4

ScrumAI in Action: A Real-World Case Study

Introduction

In this final chapter, we will present a case study of developing a real intelligent system for predicting store sales while adopting the ScrumAI methodology proposed in the previous chapter. We will begin by outlining the problematic scenario that necessitated the development of this intelligent system and the solution approach we chose to address it.

Next, we will discuss a detailed exploration of Sprint Zero, which represents the initial phase in the ScrumAI process. This will provide insight about the requirements of the client, the development plan, the selection of roles and development team, the cost of the project, the technology to use, and other crucial groundwork laid during this crucial starting point of the project.

Following this, we will briefly discuss the remaining development sprints, offering a high-level overview of how the project progressed through the iterative nature of the ScrumAI framework.

Finally, we will conclude the chapter by reflecting on the positive outcomes as well as any challenges or limitations encountered during the application of the ScrumAI methodology to this real-world intelligent system development initiative. This will serve to further validate the practical applicability and effectiveness of ScrumAI.

4.1 Business need

Sales and product demand forecasts are very necessary to brick-and-mortar grocery stores, which must carefully navigate how much inventory to buy. Predict a little over, and grocers are stuck with overstocked, perishable goods. Guess a little under, and popular items quickly sell out, leading to lost revenue and upset customers. More accurate forecasting can decrease food waste related to overstocking and improve customer satisfaction by having just enough of the right products at the right time. This challenge of striking the right balance between supply and demand is a critical problem facing many grocery retailers.

4.2 The solution

There exist a number of techniques for demand forecasting by taking advantage of past sales data. Some use Excel sheets to draw special graphs and analyze them manually in an attempt to predict the future. These graphs include the Trend graph, which is a smooth graph that captures the general behavior of sales over a long period of time (3 months or longer) [42], [43]. However, this graph is not good at capturing seasonal (short time) behaviors of your sales. The Seasonality graph is another type of graph that takes seasonality into account and can capture weekly or monthly behaviors of sales [42], [42]. Some grocers also use their long experience in the domain to predict future demand.

Today, new techniques have emerged that are based on AI algorithms and techniques, more precisely machine learning techniques, which will be our go-to solution for this problem.

Hence, our proposed solution will be based on AI techniques where we will be building a machine learning model that can predict future sales. The model will be trained on past sales data, as well as taking into consideration the trend, seasonality, and other important factors by creating new columns and data that contains those considerations such as the trend and seasonality. This considerations could have a high impact on forecasting.

Finally, we will be integrating this machine learning model into a user-friendly interface accompanied with a number of easy-to-understand graphs in order to get a complete intelligent system or what's known as a decision support system. This will allow grocery stores to leverage the power of AI and data-driven insights to optimize their inventory management and improve customer satisfaction.

4.3 Case study

In this section, we will be adopting the ScrumAI methodology to develop the solution proposed before

4.3.1 Sprint zero

This is the first stage of the ScrumAI process, it is crucial for establishing the project's explicit requirements (both functional and non-functional), its duration, its cost, the roles involved, the tools to use, and the creation of a development plan.

4.3.2 Functional requirements

Functional requirements describe the functionalities and the behavior of our decision support system, which include two main functionalities:

Forecasting Functionalities:

- The system should generate accurate sales forecasts for a chosen store and product family at a specified future date.
- The system should provide intuitive data visualizations, such as graphs and charts, to clearly communicate the sales forecast. These visualizations should make it easy for users to understand the predicted sales trends and patterns.
- In addition to the sales forecast, the system should provide supplementary information that can aid decision-making such as.

Dashboard Functionalities:

- The system should provide interactive dashboards that allow users to explore historical sales data, display visual representation of past data, including trends of sales, seasonalities, best-selling product families, and best-selling stores.
- This should include visualizations such as line charts for sales trends and seasonality patterns, and bar charts for top-performing product families and stores.

4.3.3 Non-functional requirements

Security:

- The system must have robust security measures to protect against unauthorized access and data breaches.
- Implement best practices for user authentication, access controls, and data encryption.

Performance:

- The system should provide quick response times for sales forecasting and data visualization.

- Ensure the system can handle the expected data volume and user load without degradation in performance.

Extensibility:

- The system architecture should be designed to accommodate future feature additions and scalability.
- Modularity and loose coupling of components will enable easy integration of new functionalities.

Model Risk Assessment:

- Implement rigorous processes to evaluate the accuracy and reliability of the machine learning models used for sales forecasting.
- Regularly monitor model performance and update them as needed to maintain prediction quality.
- Identify any potential business risks associated with the use of the intelligent system.

4.3.4 Roles

The ScrumAI project requires seven distinct roles to ensure a fluid workflow. Typically, these roles are defined after the product backlog and the number of sprints have been established. However, in our case, there are only three people available. Given this constraint, the roles can be assigned directly, without first defining the product backlog and sprint details (see Table 6).

Note: we did not have access to a domain expert and a cognitive scientist therefore we did not assignee them to anyone.

Roles	MR Achour Achroufene	Moh Mohamed	Zergoun Yasser
Client	√	.	.
Product Owner	√	.	.

Scrum Master	√	.	.
Engineers	.	√	√
Data Scientists	.	√	√

Table 6 - Distribution of roles

4.3.5 Product backlog

After the meeting with the client and identifying both the functional and non-functional requirements, the product owner creates a product backlog that contains all the features that need to be implemented in order to fulfill the client's requirements (see Table 7).

id	User stories
1	As a user, I can input data for a specific store and product family to generate a sales forecast for a future date.
2	As a user, I want to see intuitive data visualizations (e.g., graphs, charts) that clearly communicate the sales forecast and predicted trends.
3	As a user, I want to see supplementary information alongside the sales forecast, such as factors influencing the prediction, confidence intervals, and potential risks/opportunities.
4	As a user, I can explore historical sales data through an interactive dashboard.
5	As a user, I want to see visual representations of past sales trends, seasonality patterns, best-selling product families, and best-selling

	stores.
6	As a user, I want the dashboard to include visualizations like line charts, heat maps, and bar charts to enhance my understanding of past performance.
7	As a user, I expect the system to provide quick response times for sales forecasting and data visualization.
8	As a user, I want the system to be able to accommodate future feature additions and scalability requirements.
9	As a user, I want to be confident in the accuracy and reliability of the machine learning models used for sales forecasting.
10	As a user, I want the system to regularly monitor and update the forecasting models to maintain prediction quality.
11	As a user, I want to understand the potential impact of the system on operations, financial performance, and regulatory compliance.

Table 7 - Product backlog

4.3.6 Tools and technology

Tkinter

Tkinter is a standard Python GUI (Graphical User Interface) library that provides a set of tools and widgets to create desktop applications with graphical interfaces. It is included with most Python installations, making it easily accessible for developers who want to build GUI applications without requiring additional installations or libraries



[44].

```
# Labels and Inputs
date_inp = DateEntry(main_frame, date_pattern="yyyy-mm-dd")
Family_Liste = ['BEVERAGES', 'BREAD/BAKERY', 'CLEANING', 'DAIRY', 'DELI', 'EGGS', 'FROZEN FOODS', 'GROCERY I', 'HOME CARE', 'LIQUOR,WINE,BEER',
value_inside = tk.StringVar(main_frame)
value_inside.set("Select a family")
Family_inp = OptionMenu(main_frame, value_inside, *Family_Liste)
onpromotion_inp = Spinbox(main_frame, from_=0, to=100, increment=1)
predict_button = Button(main_frame, text="Predict", command=get_input)

date_inp_l = Label(main_frame, text="Date", font=("Arial", 17), anchor="w")
Family_inp_l = Label(main_frame, text="Product Family", font=("Arial", 17), anchor="w")
onpromotion_inp_l = Label(main_frame, text="Number Of Promotions ", font=("Arial", 17), anchor="w")
result = Label(main_frame, text="", font=("Arial", 17))

date_inp_l.grid(row=0, column=0, sticky="ew", padx=5, pady=5)
date_inp.grid(row=0, column=1, sticky="ew", padx=5, pady=5)
Family_inp_l.grid(row=1, column=0, sticky="ew", padx=5, pady=5)
Family_inp.grid(row=1, column=1, sticky="ew", padx=5, pady=5)
onpromotion_inp_l.grid(row=2, column=0, sticky="ew", padx=5, pady=5)
onpromotion_inp.grid(row=2, column=1, sticky="ew", padx=5, pady=5)
predict_button.grid(row=3, column=1, sticky="ew", padx=5, pady=5)
result.grid(row=4, column=1, sticky="ew", padx=5, pady=5)

root.mainloop()
```

Figure 17 - Code snippet using Tkinter to create user interface

Pandas

Pandas is a Python package that provides fast, flexible, and expressive data structures designed to make working with "relational" or "labeled" data both easy and intuitive. It aims to be the fundamental high-level building block for doing practical, real world data analysis in Python. Additionally, it has the broader goal of becoming the most powerful and flexible open source data analysis / manipulation tool available in any language. It is already well on its way towards this goal [45].



```
18     if strnmbr=='44' :
19         def load_dataset():
20             train_data = pd.read_csv('train_data5.csv',dtype={
21                 'sales': 'float32',
22                 'onpromotion': 'int32'},
23                 parse_dates=['date'])
24             return train_data
```

Figure 18 - Code snippet using pandas to load a dataset

Scikit-learn

Scikit-learn, also known as sklearn, is an open-source, machine learning and data modeling library for Python. It features various classification, regression and clustering algorithms including support vector machines, random forests, gradient boosting, k-means and DBSCAN, and is designed to interoperate with the Python libraries, NumPy and SciPy. It implements numerous data modeling and machine learning algorithms, and provides consistent Python APIs. It supports a standardized and concise model interface across models. For example, Scikit-learn makes use of a simple fit/predict workflow model for its classification algorithms [46].



Matplotlib

Matplotlib is an open-source Python library used to create high-quality graphs and charts and serves as an open-source alternative to MATLAB. For instance, you can create plots, histograms, bar charts, and various types of graphs with just a few lines of code. It's a comprehensive tool that enables the generation of highly detailed data visualizations. This library is especially



valuable for individuals working with Python or NumPy [47].

```
103 fig_prd, ax_prd = plt.subplots(figsize=(7, 4.4))
104 ax_prd.plot(list(period_prediction.index) , list(period_prediction['sales']))
105 ax_prd.set_title('Sales Trend')
106 ax_prd.set_xlabel('Number of days')
107 ax_prd.set_ylabel('Sales')
108 ax_prd.set_xlim(0,max(list(period_prediction.index)))
109 plt.subplots_adjust(bottom=0.2)
```

Figure 19 - code snippet using matplotlib to display graphs

NumPy

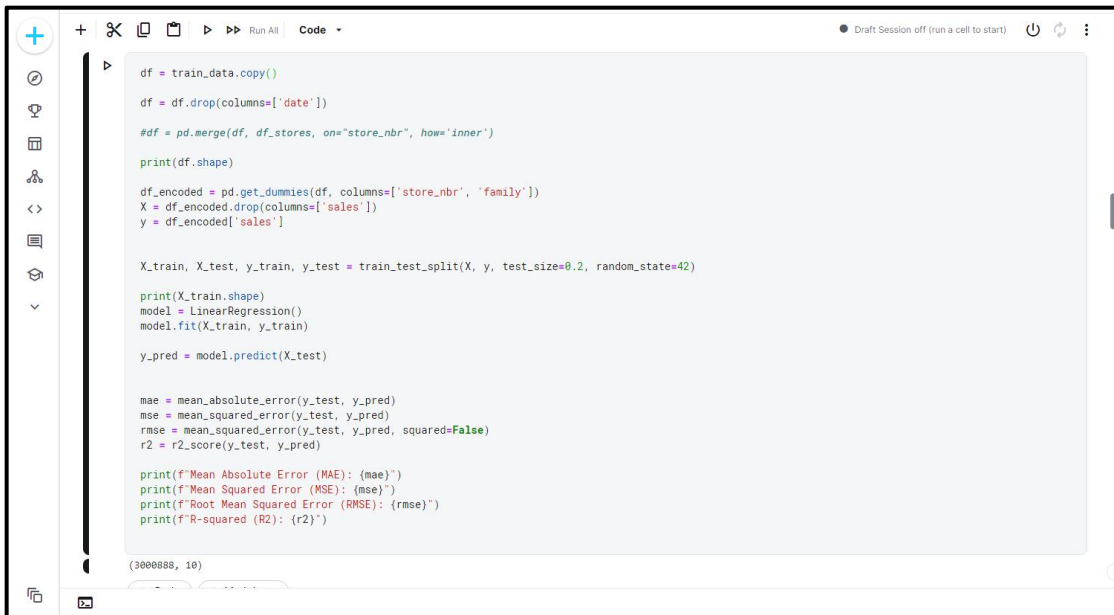
NumPy is a python library used to work with arrays. It also contains functions for working with matrices, the Fourier transform, and linear algebra. Free to use it as it is an open-source project [48].



Kaggle

Kaggle serves as a platform for data science competitions, where machine learning engineers and data scientists can compete to develop the best models for analyzing particular data sets or addressing particular issues. Additionally, the platform offers a community where users may exchange code and data sets, work together on projects, and benefit from one another's expertise. Kaggle was founded in 2010 and is currently a part of Google Cloud. Google acquired the platform in 2017 [49].





```
df = train_data.copy()
df = df.drop(columns=['date'])

#df = pd.merge(df, df_stores, on="store_nbr", how="inner")

print(df.shape)

df_encoded = pd.get_dummies(df, columns=['store_nbr', 'family'])
X = df_encoded.drop(columns=['sales'])
y = df_encoded['sales']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

print(X_train.shape)
model = LinearRegression()
model.fit(X_train, y_train)

y_pred = model.predict(X_test)

mae = mean_absolute_error(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
rmse = mean_squared_error(y_test, y_pred, squared=False)
r2 = r2_score(y_test, y_pred)

print(f"Mean Absolute Error (MAE): {mae}")
print(f"Mean Squared Error (MSE): {mse}")
print(f"Root Mean Squared Error (RMSE): {rmse}")
print(f"R-squared (R2): {r2}")
```

Figure 20 - Code snippet from kaggle environment to train the machine learning model

4.4 Sprint 1

Sprint 1 represents the first iteration in the development process. After the sprint planning meeting where the development team with the product owner discusses which features from the product backlog should be implemented during this sprint, as well as the time estimates for each feature and the distribution of responsibilities among the team members. A Sprint Backlog is created with priorities assigned from top to bottom as shown in Table 8.

PRIORITY	USER STORY	ASSIGNED	TIME
		TO	

1	As a user, I can input data for a specific store and product family to generate a sales forecast for a future date.	Moh Mohamed	2 weeks
2	As a user, I want to see intuitive data visualizations (e.g., graphs, charts) that clearly communicate the sales forecast and predicted trends.	Moh Mohamed	4 days
3	As a user, I want to see supplementary information alongside the sales forecast, such as factors influencing the prediction, confidence intervals, and potential risks/opportunities.	Zergoun Yasser	3 days

Table 8 - Sprint 01 backlog

The Sprint Backlog includes the following information:

- Id: a unique identifier for each user story.
- User story: a description of the user's need or requirement.
- Assigned to: the member of the development team responsible for implementing the user story.

This sprint backlog represents the first set of features to be implemented, as determined by the team's priorities and the client's requirements.

4.4.1 Development

In the ScrumAI methodology, the development process is composed of two parallel cycles:

A software cycle, which concerns everything related to digitalization, user interface, and automating tasks like adding or deleting products from a database.

A model cycle, which concerns developing a machine learning model.

The distribution of work can vary depending on the team size, the developers' competencies, and their preferences. The team can choose to work on the two cycles (software cycle and model cycle) in parallel, with developers assigned to different tracks. Alternatively, the team can focus on one cycle at a time.

In our case, the team has only two developers, thus we have decided to work on one cycle at a time, rather than splitting the work across the two tracks.

4.4.1.1 Model cycle

Since the first priority in the Sprint Backlog is the Sales Forecasting feature, the team has decided to start by focusing on the model cycle. This involves the following steps:

Data acquisition

This step involves gathering the necessary data required for training the machine learning model, such as historical sales data, product information, and any other relevant data sources.

In a real-world scenario, these datasets would typically belong to the client's stores. The involvement of a domain expert and a cognitive scientist in this step is crucial, where the domain expert can help with identifying high-quality data that may be needed, while the cognitive scientist is responsible for communicating with the client and gather the necessary data, as well reviewing the data and remove any sensitive information that the client would not want to be viewed. These two roles will help ensure high-quality data while preserving the security of these data.

However, in our case, our supervisor is simulating the client, and we do not have access to an actual dataset as well as the lack of both the domain expert and a cognitive scientist.

Given this constraint, we had to look for a public dataset that has already been reviewed and processed by a domain expert and a cognitive scientist. Therefore, we explored the Kaggle platform and found a relevant dataset that serves our purpose.

The "Store Sales - Time Series Forecasting" dataset belongs to the Favorita stores located in Ecuador. The dataset records the daily sales over a 5-year period from 2013 to 2017, resulting in a total of 3 million rows of data. The dataset includes 6 attributes [50]:

- Id
- Daily time-series data
- Product family (with 33 unique values)
- Store number (with 55 unique values)
- The total number of promotions of a given product family at a given date and store number
- Sales, which represents the total sales at a given date, store number, and product family

In addition to the sales data, the dataset also includes other relevant information, such as: Holidays dataset, Daily oil price dataset, Transactions dataset, Store details data

This comprehensive dataset provides a valuable opportunity to develop a forecasting model for store sales.

Data processing

In this phase, we prepare and transform the data to make it ready for model building.

This may involve tasks such as handling missing values, dealing with outliers, feature engineering, and other data transformation activities. Data processing may also involve the domain expert's knowledge and guidance in order to select the appropriate transformations to apply to the dataset.

In our case, since we did not have access to an actual domain expert to guide the data transformation phase, we used the resources listed in the dataset description [42] in addition to online research to look for information about the domain. This help us detect and understand what are the important factors and attribute that should be taken in consideration for sales forecasting.

We first checked for any null values in the dataset. We then eliminated the 'id' attribute, as it did not provide any meaningful information for the modeling process.

Next, we focused on feature engineering. This involved transforming the date-time series into more useful features. We extracted the day of the month, day of the week, month, year, and the new year separately. This will allow us to capture any seasonal patterns in the data.

On top of that, we extracted another feature known as 'Lag'. Lagging a time series means to shift its values forward one or more time steps [42], for example if you have a daily time series, the 'lag_1' feature would have the values from one day ago, the 'lag_2' feature would have values from two days ago, and so on (see Figure 21).

The basic idea behind using lag features is that the current value of a time series often depends on its past values. For this reason we used not only the lag_1 feature but also we considered the mean value of lag_7, this will allow us to capture information about the whole previous week.

	y	y_lag_1	y_lag_2
Date			
1954-07	5.8	NaN	NaN
1954-08	6.0	5.8	NaN
1954-09	6.1	6.0	5.8
1954-10	5.7	6.1	6.0
1954-11	5.3	5.7	6.1

Figure 21 - Lag features

In the dataset description they also mentioned that a magnitude 7.8 earthquake struck Ecuador on April 16, 2016, resulting in people buying essential needs and donating them [50], therefore we added this information as a separate column or feature that might capture some pattern in the dataset.

Additionally, we created a 'trend' feature by calculating the mean value of sales of a given store number and a product family over a one-month period. This feature can help us understand the overall sales trend over time.

The 'store number' and 'product family' columns are categorical in nature, so we transformed them into numerical representations to make them suitable for modeling.

Finally, we split the dataset into two parts: a training set and a test set. The training set will be used to train the machine learning model, while the test set will be used to evaluate the model's performance using various metrics.

By performing these data preprocessing and transformation steps, we have prepared the dataset to be effectively utilized for model building and evaluation.

Model building and evaluation

The model building phase involves several key steps: Selecting the appropriate model, tuning hyperparameters, and Training the model.

Since we are dealing with a supervised regression problem, we chose a 'LinearRegression' as the base model. We trained the model and calculated a number of evaluation metrics, which include: Mean Absolute Error (MAE), Mean Squared Error (MSE), Root Mean Squared Error (RMSE), Root Mean Squared Logarithmic Error (RMSLE), R-squared (R²).

The results of these evaluations are presented in Table 9.

ERROR	VALUE
MEAN ABSOLUTE ERROR	107.42
MEAN SQUARED ERROR	137238.24
ROOT MEAN SQUARED ERROR	370.45
ROOT MEAN SQUARED LOGARITHMIC ERROR	nan
R-SQUARED	0.88

Table 9 - Evaluation metrics (model 1)

The base LinearRegression model has performed reasonably well, with an accuracy of 88% and an error value of approximately 370.45 sales units. This suggests the model is making accurate sales predictions, but there is still a room for improvement.

In the upcoming Sprint, we will experiment with different models and processing techniques to see if they can further enhance the model's performance and reduce the error rate.

Sprint deployment

Once the model is built and validated, this phase include exporting the model to be integrated with the application.

By focusing on the model life cycle first, the team managed to implement the core machine learning aspects of the forecasting functionalities before moving on to the software life cycle tasks, such as the user interface and data visualization components.

4.4.1.2 Software cycle

Once we have implemented the model, the next step is to integrate it with a user interface that has a number of input fields and a way of displaying the forecasting result in the form of text as well as graph visualization.

In the software development cycle, the requirements have already been defined, which leaves us with three main phases to focus on:

Architecture design or conception

- In this phase, the developers design the necessary architectural diagrams, including:
- Use Case Diagrams that describe the different functionalities with their appropriate actors and their access rights.
- Sequence diagram, which details the order in which the object interact including: Lifetime of the object, the processes that interact with it and the messages exchanged.
- Class Diagrams that describe the various entities within the application and serve as a template for building the database.

In our case, as the main goal is to create an intelligent system, which does not have any database-related functionalities, thereby we are utilizing only a Use Case Diagram, as shown in Figure 22. The user can input data to predict sales, visualize the prediction through a graph, and get useful information depending on the input.

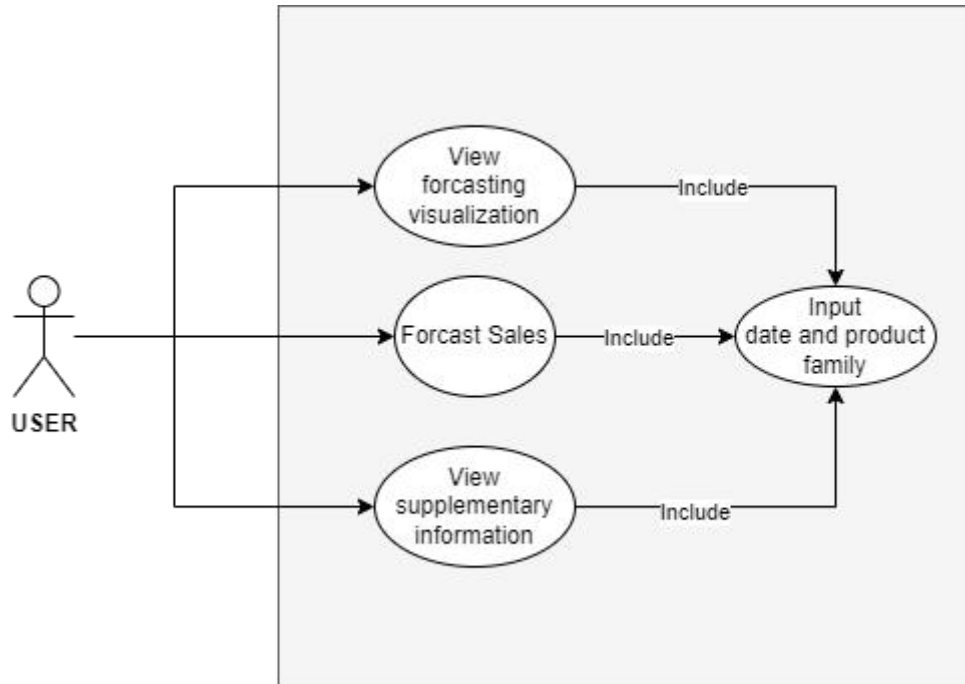


Figure 22 - Use case diagram (sprint 1)

Implementation

This is the coding phase, where the team implements the required functionalities and user interfaces. We have used the Python library Tkinter for the interface and other necessary libraries such as Matplotlib for plotting the graphs, as well as Pickle for loading the machine learning model and integrating it with the interface.

Test and sprint deployment

In this phase, the team thoroughly tests the developed features and fixes any identified bugs before delivering the final build of Sprint 1 to the client for testing.

ScrumAI emphasizes creating a brief model documentation at the end of each sprint, detailing the model development process. Additionally, they emphasize using a data/model version control tool to keep track of model experiments.

Table 10 provides an example of the model documentation, highlighting the data processing steps, model architecture, model parameters, and the evaluation metrics used.

<p>Data processing steps</p>	<ul style="list-style-type: none"> • Eliminated the 'id' attribute, as it did not provide meaningful information. • Created features for the day, month, and year to capture seasonal patterns. • Created a 'lag 1' feature and the mean of 'lag 7' to capture dependencies on past values. • Created a 'trend' feature by calculating the mean sales over a one-month period. • Added a feature to capture the impact of the magnitude 7.8 earthquake in Ecuador. • Transformed the 'store number' and 'product family' columns into numerical representations. • Split the dataset into training and test sets.
<p>Model architecture, model parameters</p>	<p>LinearRegression with no special hyperparameters</p>
<p>Evaluation metrics</p>	<ul style="list-style-type: none"> • Mean Absolute Error (MAE). • Mean Squared Error (MSE). • Root Mean Squared Error (RMSE). • Root Mean Squared Logarithmic Error (RMSLE). • R-squared (R²).

Table 10 - Model documentation (sprint 1)

As for data/model version control, since the team did not have enough time and experience with specialized data/model version control tools, we opted for a more simplistic approach. We used a spreadsheet to store an instance of the data used, and exported the model as a file in the same directory as the dataset instance (see Figure 23).

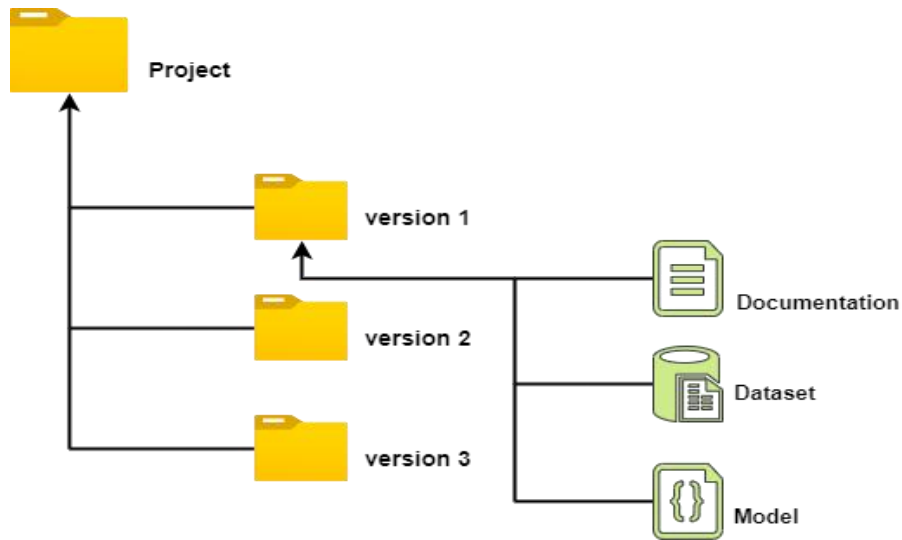


Figure 23 - Model/Data version control example

Finally, the output of this sprint shown in Figure 24 is a sprint release that contains a number of working functionalities including predicting sales, visualizing the result, and displaying useful information. To wrap up this sprint, a retrospective meeting is scheduled to discuss what went well and what could be improved for the next sprint.

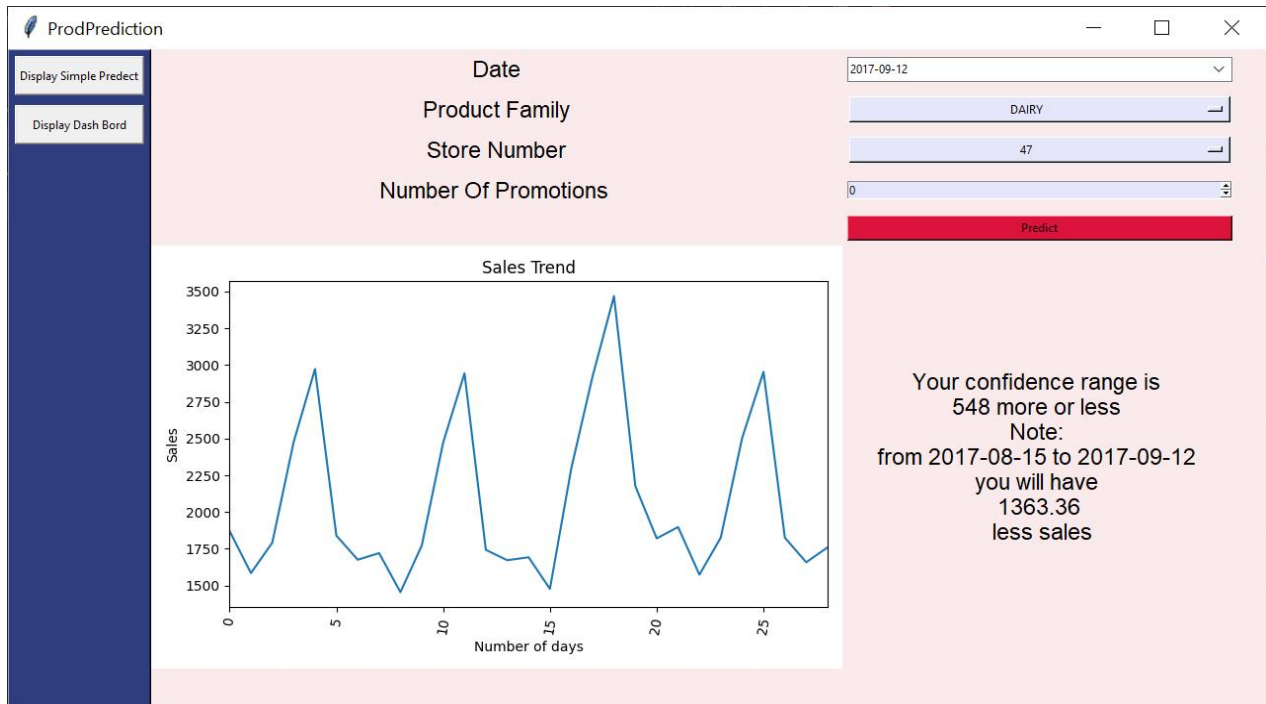


Figure 24 - Forecasting interface

4.5 Sprint 2

After the second sprint meeting, another sprint backlog was defined as shown in Table 11:

PRIORITY	USER STORY	ASSIGNED TO	TIME
1	As a user, I can explore historical sales data through an interactive dashboard.	Moh Mohamed	1 week
2	As a user, I want to see visual representations of past sales trends, seasonality patterns, best-selling	Moh Mohamed	4 days

	product families, and best-selling stores.		
3	As a user, I want the dashboard to include visualizations like line charts, heat maps, and bar charts to enhance my understanding of past performance.	Zergoun Yasser	3 days

Table 11 - Sprint 02 backlog

4.5.1 Development

In the second sprint, our goal was to implement a dashboard and improve the model forecasting. This time, we started with software-related features rather than the model, as we believed that improving the model after creating a base model would be relatively easier and would not take a lot of time, as most of the groundwork had been done.

4.5.1.1 Software cycle

Following the same procedures as Sprint 1, we modeled a use case diagram as shown in Figure 25, then implemented the different functionalities, tested them, and fixed any bugs that were found, as well as bugs that were raised by the client from the previous build.

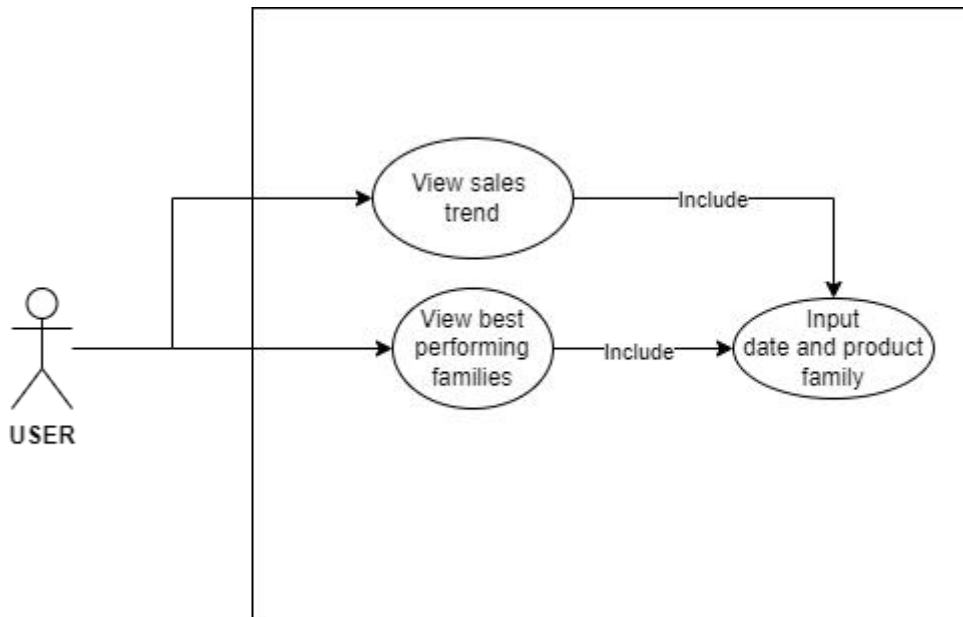


Figure 25 - Use case diagram (sprint2)

4.5.1.2 Model cycle

In this model lifecycle, we tried the RandomForestRegressor model. However, we faced an issue when training the model because the dataset was very large, with 3 million rows and 96 columns (created from the conversion of categorical columns to numerical columns). Since we did not have the required resources to train this large dataset, we had to use different processing techniques this time.

Data processing

After visualizing the data and presenting the performance of each store number and product family, we selected the top performing store (store number 44), and then we selected the top 16 performing product families for every store, as shown in Figure 26, Figure 27.

This resulted in a significant decrease in the dataset size. Following this, we applied the same data processing steps as the previous sprint, including seasonality features, trend feature; lag features, and converting categorical features to numerical. Finally, we split the data into training and test sets to train the model.

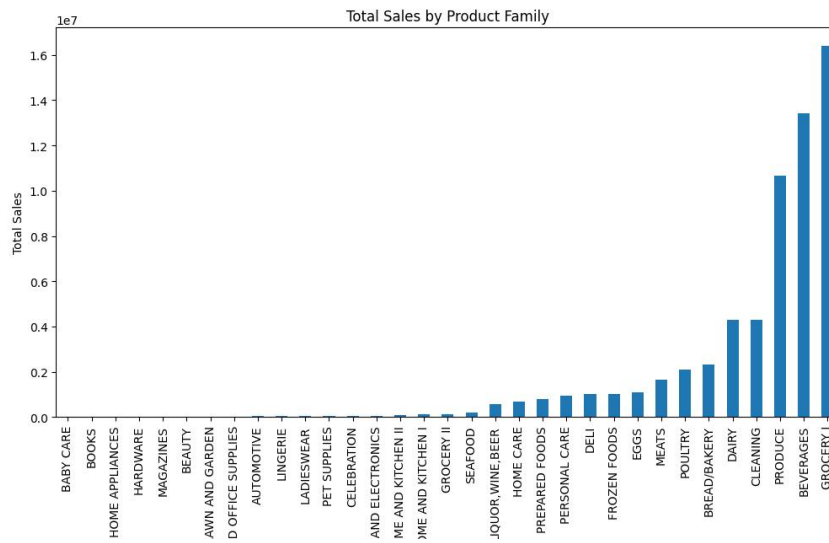


Figure 26 - Most selling product families

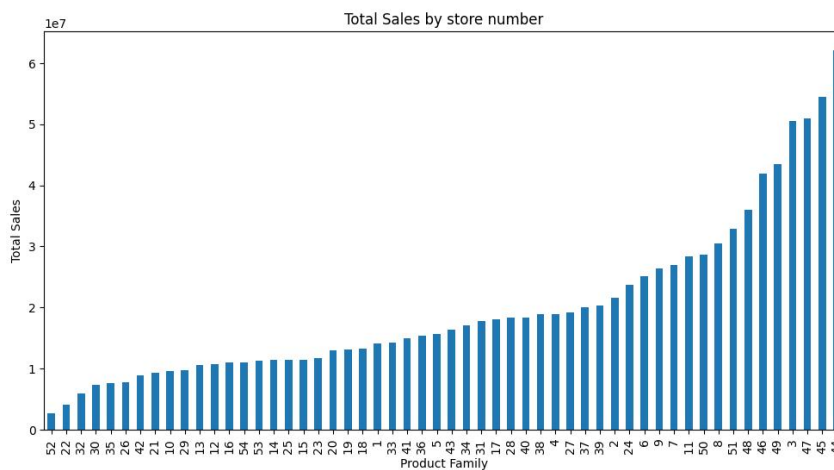


Figure 27 - Most selling stores

Model building and evaluation

We trained the RandomForestRegressor model with a dataset of store 44 only, and the following hyperparameters:

- The number of decision trees that will be used in the Random Forest model: `n_estimators = 100`
- the model's random number generator: `random_state = 42`

For the evaluation, we used the same metrics as the previous sprint. The results are presented in Table 12 in addition to a comparison to the previous model (model 1).

ERROR		MODEL 1	MODEL 2
		LINEAR REGRESSOR	RANDOM FOREST REGRESSOR
MEAN ERROR	ABSOLUTE	107.42	256.11
MEAN ERROR	SQUARED	137238.24	428406.22
ROOT SQUARED ERROR	MEAN	370.45	654.52
ROOT SQUARED LOGARITHMIC ERROR	MEAN	nan	0.66
R-SQUARED		0.88	0.96

Table 12 - Evaluation metrics (model 2) and a comparison between the two models

Sprint deployment

Since we now have two different models to work with, we would need to choose the best one to deploy to our application. To do this, we will use the comparative table to evaluate the two models based on the different evaluation metrics (see Table 12). Additionally, an expert in the domain should be consulted, as the evaluation metrics can depend on the specific requirements of the domain.

While Model 1 has lower individual error metrics, the significantly higher R-squared (Accuracy) of Model 2 indicates that it is likely a better choice for sales forecasting. The higher R-squared means Model 2 is better at capturing most of the variability of sales, which is critical for producing accurate and reliable sales forecasts.

In the sales forecasting domain, the ability to explain and predict the sales trend is often more important than minimizing individual errors. The higher R-squared of Model 2 suggests it is the superior model for this specific use case, despite its slightly higher individual error metrics compared to Model 1.

Thus, we exported and deployed Model 2 to the application and we made the necessary changes and configuration to the interface code to make it work with Model 2.

To extend the application further and allow the forecasting of multiple stores, we trained four additional RandomForestRegressor models. Each model was trained on its own corresponding dataset, specific to each store. Finally, we fixed any possible bugs before giving this build to the client to test it.

A model documentation for sprint 2 (presented in Table 13) is also defined in addition a model/data instances have been saved. The sprint 2 output is shown in Figure 28, and Figure 29.

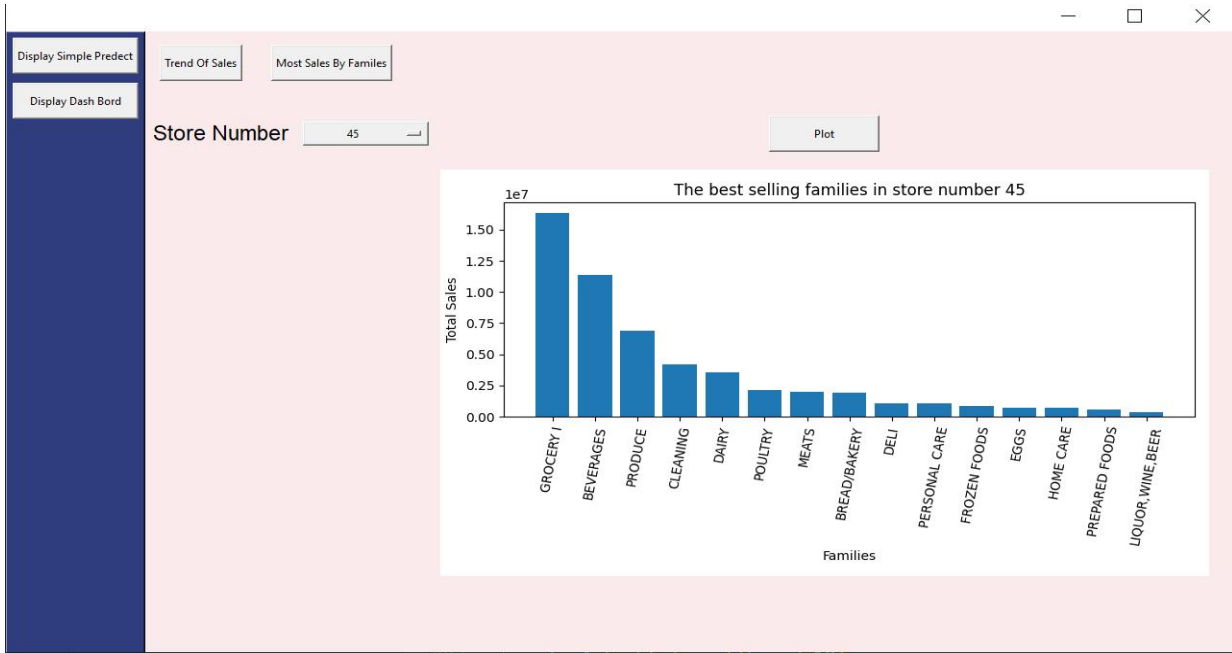


Figure 28 - Dashboard interface 1



Figure 29 – Dashboard interface 2

Data processing steps	<ul style="list-style-type: none"> • Eliminated the 'id' attribute, as it did not provide meaningful information. • Selected the sales of store number 44
-----------------------	---

	<ul style="list-style-type: none"> • Selected the sales of the top 16 performing product families • Created features for the day, month, and year to capture seasonal patterns. • Created a 'lag 1' feature and the mean of 'lag 7' to capture dependencies on past values. • Created a 'trend' feature by calculating the mean sales over a one-month period. • Added a feature to capture the impact of the magnitude 7.8 earthquake in Ecuador. • Transformed the 'store number' and 'product family' columns into numerical representations. • Split the dataset into training and test sets.
<p>Model architecture, model parameters</p>	<ul style="list-style-type: none"> • RandomForestRegressor • n_estimators =100 • random_state =42
<p>Evaluation metrics</p>	<ul style="list-style-type: none"> • Mean Absolute Error (MAE). • Mean Squared Error (MSE). • Root Mean Squared Error (RMSE). • Root Mean Squared Logarithmic Error (RMSLE). • R-squared (R2).

Table 13 - Model documentation (sprint 2)

4.6 Evaluating ScrumAI

Even though an additional final sprint (sprint 3) for optimization, code refactoring, security, and model output safety would have been ideal, time constraints prevented us from completing that final phase. However, we have achieved 90% completion of the intelligent system by following the ScrumAI workflow. This methodology, which splits the development process into two work cycles, allowed the team to focus their efforts on one phase at a time. As a result, we experienced better collaboration among team members, faster development in a shorter period, fewer bugs, and overall improved software and model quality

The comprehensive model documentation enabled the team to easily recall data processing steps, model parameters, and evaluation metrics whenever needed. This ensured traceability and transparency throughout the development process.

The data and model version control system allowed the team to store an instance of each experiment, including the training dataset, test dataset, and the trained model. This proved to be very valuable, as it gave the team the ability to easily roll back to a previous model if the client was not satisfied with the latest version. This flexibility was crucial for the project.

The integration of the cognitive scientist helped to address several issues. It fixed security problems by ensuring that the client's dataset was only accessible to the cognitive scientist, which would then communicate a sanitized version to the development team. Additionally, the cognitive scientist provides a well-structured dataset that is easily understandable by the development team

The domain expert played a crucial role in data processing, suggesting useful features that could impact the forecasting as well recommending specific metrics for the model evaluation process and interpreting the results based on their expertise.

Overall, this robust ScrumAI workflow enabled the development team to work efficiently, deliver high-quality models, and address important problems specific to AI application. The key benefits include: improved team collaboration, faster development, reduced bugs, traceability, flexibility, and the incorporation of domain knowledge.

Conclusion

In this chapter, we have explored a case study example of developing an intelligent decision support system using the ScrumAI methodology. We have delved into the different phases of ScrumAI in detail, within a real-world scenario, to test its effectiveness and identify potential challenges.

Despite having a small development team of only two developers with a limited time frame, ScrumAI proved to be effective in delivering high-quality intelligent applications.

However, to fully leverage the benefits of ScrumAI, which involves two distinct cycles, the development team requires experience in both cycles including software engineering and model building/machine learning. ScrumAI also could result in difficulties for managing the many different roles.

General conclusion

The main objective of this thesis was to propose a methodology for the development of intelligent systems, taking into account the specification of the process of building applications integrating AI and machine learning techniques. To achieve this, we first explored intelligent systems and realized that their role is essential in business, especially in decision-making and recommendation. In addition, intelligent systems make use of machine learning techniques for extracting and processing complex data, knowledge acquisition and representation techniques for building knowledge bases, and improving the quality of decisions. Next, we looked at software development methods and found that there are several types, with classical methods showing their limits in terms of flexibility and adaptability to change, in contrast to agile methods, which have emerged as an effective response thanks to their flexible, iterative approach.

Furthermore, we have delved into the state of the art of development methodologies for AI systems, highlighting the challenges and specifications of these systems. Through these studies, it has become evident that existing methodologies are not suitable for AI development, and there is a pressing need for a structured methodology that can effectively handle the unique specifications and challenges of AI systems development such as data acquisition and processing, the integration of new roles like domain experts, the evolving nature of intelligent systems with continuous experiments, collaboration between data scientists and developers, and the emergence of new non-functional requirements, raise questions about the sufficiency of existing approaches. In response to this need, this work has presented a new methodology called ScrumAI, an adaptation of the Scrum framework tailored to meet the expectations of AI systems. It adds a model sprint for learning and building model phase of the development of intelligent systems including new actors as data scientist and domain expert, while benefiting from the flexibility of agile methods.

To demonstrate the effectiveness of ScrumAI, a case study was conducted for developing a decision support system that can forecast store sales and generate useful graphs. We detailed the workflow of ScrumAI, including Sprint Zero where requirements specification and planification were defined, Sprint 1 for developing the main functionalities, and a general overview of Sprint 2 for the rest of the functionalities.

General conclusion

This case study resulted in a fully functional intelligent system containing a machine learning model with high accuracy to predicate accurate sales, in addition the systems display a visual interface for smart graph representations and other useful information and software related functionalities, this proves the effectiveness of ScrumAI in building a complete intelligent system.

As we look to the future, there is still much room for improvement in AI development methodologies. One potential avenue for exploration is the integration of Machine learning operations (MLOps) best practices and rules to further enhance the efficiency and effectiveness of ScrumAI. Moreover, we envision a future where AI techniques and applications are integrated in the development methodologies, to augment and support them. This could include the development of intelligent techniques and tools for automatic documentation, intelligent management and work distribution, intelligent agents simulating domain experts, automatic data processing, testing, model selection, and training. Such integration has the potential to revolutionize the development process, enabling developers to focus on higher-level creative tasks while AI systems handle routine and repetitive tasks. Therefore, we propose that future research should focus on refining and expanding ScrumAI to address the emerging challenges and opportunities in AI development, and exploring the potential of AI-augmented development methodologies.

References

- [1] J. Schuett, "A Legal Definition of AI," *SSRN Electron. J.*, 2019, doi: 10.2139/ssrn.3453632.
- [2] A. M. Turing, "Computing Machinery and Intelligence," in *Parsing the Turing Test: Philosophical and Methodological Issues in the Quest for the Thinking Computer*, R. Epstein, G. Roberts, and G. Beber, Eds., Dordrecht: Springer Netherlands, 2009, pp. 23–65. doi: 10.1007/978-1-4020-6710-5_3.
- [3] J. McCarthy, "What is Artificial Intelligence?," Jan. 2004.
- [4] M. Molina, "What is an intelligent system?" arXiv, Dec. 18, 2022. Accessed: Jun. 23, 2024. [Online]. Available: <http://arxiv.org/abs/2009.09083>
- [5] S. Martínez-Fernández *et al.*, "Software Engineering for AI-Based Systems: A Survey," *ACM Trans. Softw. Eng. Methodol.*, vol. 31, no. 2, pp. 1–59, Apr. 2022, doi: 10.1145/3487043.
- [6] S. LAGAB and K. GALI, "Introduction de l'agilité dans le processus de développement de systèmes intelligents," University of bejaia, Bejaia, 2023. [Online]. Available: <https://www.univ-bejaia.dz/xmlui/bitstream/handle/123456789/23179/PFE.pdf?sequence=1&isAllowed=y>
- [7] S. J. Russell, P. Norvig, and E. Davis, *Artificial intelligence: a modern approach*, 3rd ed. in Prentice Hall series in artificial intelligence. Upper Saddle River: Prentice Hall, 2010.
- [8] "What is a Recommendation System?," NVIDIA Data Science Glossary. Accessed: Jun. 11, 2024. [Online]. Available: <https://www.nvidia.com/en-us/glossary/recommendation-system/>
- [9] "What is a decision support system (DSS)?," CIO. Accessed: Jun. 11, 2024. [Online]. Available: <https://www.techtarget.com/searchcio/definition/decision-support-system>
- [10] "Expert system | AI, Knowledge Representation & Reasoning | Britannica." Accessed: Jun. 11, 2024. [Online]. Available: <https://www.britannica.com/technology/expert-system>

References

- [11] Y. Cao *et al.*, *A Comprehensive Survey of AI-Generated Content (AIGC): A History of Generative AI from GAN to ChatGPT*. 2023.
- [12] A. Stavridis and A. Drugge, “THE RISE OF INTELLIGENT SYSTEM DEVELOPMENT,” Umeå University, Faculty of Social Sciences, Department of Informatics., 2023. [Online]. Available: [urn:nbn:se:umu:diva-208456](https://nbn-resolving.org/urn:nbn:se:umu:diva-208456)
- [13] W. Westera *et al.*, “Artificial intelligence moving serious gaming: Presenting reusable game AI components,” *Educ. Inf. Technol.*, vol. 25, no. 1, pp. 351–380, Jan. 2020, doi: 10.1007/s10639-019-09968-2.
- [14] “Applications of artificial intelligence (AI),” Google Cloud. Accessed: Jun. 11, 2024. [Online]. Available: <https://cloud.google.com/discover/ai-applications>
- [15] “Top 20 Applications of Artificial Intelligence (AI) in 2024,” GeeksforGeeks. Accessed: Jun. 11, 2024. [Online]. Available: <https://www.geeksforgeeks.org/applications-of-ai/>
- [16] P. Buxmann, T. Hess, and J. B. Thatcher, “AI-Based Information Systems,” *Bus. Inf. Syst. Eng.*, vol. 63, no. 1, pp. 1–4, Feb. 2021, doi: 10.1007/s12599-020-00675-8.
- [17] “FIGURE 1. Artificial Intelligence Life Cycle.,” ResearchGate. Accessed: Jul. 09, 2024. [Online]. Available: https://www.researchgate.net/figure/Artificial-Intelligence-Life-Cycle_fig1_371713457
- [18] L. Khong, L. Yu Beng, T. Yip, and T. Soofun, “Software Development Life Cycle AGILE vs Traditional Approaches,” Feb. 2012.
- [19] “DevOps - Resources and Tools,” IBM Developer. Accessed: Jun. 11, 2024. [Online]. Available: <https://developer.ibm.com/devpractices/devops/>
- [20] A. GUMIŃSKI, K. DOHN, and E. OLOYEDE, “ADVANTAGES AND DISADVANTAGES OF TRADITIONAL AND AGILE METHODS IN SOFTWARE DEVELOPMENT PROJECTS – CASE STUDY,” Silesian University of Technology, 2023.
- [21] J. Highsmith, “Agile Project Management: Creating Innovative Products,” Mar. 2004.
- [22] A. El Mehdi, C. M. Yassin, and E. K. K. Eddine, “Survey and Comparative Study on Agile Methods in Software Engineering,” *Trans. Mach. Learn. Artif. Intell.*, vol. 5, no. 4, Aug. 2017, doi: 10.14738/tmlai.54.3203.

References

- [23] M. Himmat and A. S. A. Osman, "AGILE SOFTWARE DEVELOPMENT METHODS AND CURRENT TRENDS," 2020.
- [24] "12 Key Agile Manifesto Principles | LeanWisdom." Accessed: Jun. 12, 2024. [Online]. Available: <https://www.leanwisdom.com/blog/agile-manifesto-principles>
- [25] B. Jean-Louis, "Extreme Programming-Agile Methods," *Tour Horison Business Interactif*, 2011.
- [26] A. Anwar, "A Review of RUP (Rational Unified Process)," *Int. J. Softw. Eng.*, 2014.
- [27] J. Sutherland and S. Ken, "The Scrum Guide 2011," Oct. 2011.
- [28] K. Schwaber and J. Sutherland, "The Definitive Guide to Scrum: The Rules of the Game," Nov. 2017.
- [29] S. Sharma, D. Sarkar, and D. Gupta, "Agile Processes and Methodologies: A Conceptual Study," vol. 4, no. 05, 2012.
- [30] S. Dasgupta and V. K. Vankayala, "Developing Real Time Business Intelligence Systems the Agile Way," in *2007 1st Annual IEEE Systems Conference*, Honolulu, HI, USA: IEEE, Apr. 2007, pp. 1–7. doi: 10.1109/SYSTEMS.2007.374652.
- [31] M. Haakman, L. Cruz, H. Huijgens, and A. Van Deursen, "AI lifecycle models need to be revised: An exploratory study in Fintech," *Empir. Softw. Eng.*, vol. 26, no. 5, p. 95, Sep. 2021, doi: 10.1007/s10664-021-09993-1.
- [32] M. S. Rahman, F. Khomh, A. Hamidi, J. Cheng, G. Antoniol, and H. Washizaki, "Machine learning application development: practitioners' insights," *Softw. Qual. J.*, vol. 31, no. 4, pp. 1065–1119, Dec. 2023, doi: 10.1007/s11219-023-09621-9.
- [33] A. Messina and I. Voloshanovskiy, "Hybrid Agile Software Development for Smart Farming Application," in *Proceedings of 6th International Conference in Software Engineering for Defence Applications*, vol. 925, P. Ciancarini, M. Mazzara, A. Messina, A. Sillitti, and G. Succi, Eds., in *Advances in Intelligent Systems and Computing*, vol. 925, Cham: Springer International Publishing, 2020, pp. 198–205. doi: 10.1007/978-3-030-14687-0_18.

- [34] S. Das *et al.*, “Agile Systems Engineering in Building Complex AI Systems,” in *Engineering Artificially Intelligent Systems*, vol. 13000, W. F. Lawless, J. Llinas, D. A. Sofge, and R. Mittu, Eds., in *Lecture Notes in Computer Science*, vol. 13000, Cham: Springer International Publishing, 2021, pp. 192–208. doi: 10.1007/978-3-030-89385-9_12.
- [35] A. Vresk, I. Pihir, and M. T. Furjan, “Agile vs. Traditional Methods for Managing IT Projects - A Case Study,” 2020.
- [36] I. W. Syahputri, R. Ferdiana, and S. S. Kusumawardani, “Does System Based on Artificial Intelligence Need Software Engineering Method? Systematic Review,” in *2020 Fifth International Conference on Informatics and Computing (ICIC)*, Gorontalo, Indonesia: IEEE, Nov. 2020, pp. 1–6. doi: 10.1109/ICIC50835.2020.9288582.
- [37] A. Abdelghany, N. Darwish, Cairo University, and H. Hefni, “An Agile Methodology for Ontology Development,” *Int. J. Intell. Eng. Syst.*, vol. 12, no. 2, pp. 170–181, Apr. 2019, doi: 10.22266/ijies2019.0430.17.
- [38] P. Kurrek, F. Zoghlami, M. Jocas, M. Stoelen, and V. Salehi, “Q-Model: An Artificial Intelligence Based Methodology for the Development of Autonomous Robots,” *J. Comput. Inf. Sci. Eng.*, vol. 20, no. 6, p. 061006, Dec. 2020, doi: 10.1115/1.4046992.
- [39] M. Lourens, R. Raman, P. Vanitha, R. Singh, G. Manoharan, and M. Tiwari, “Agile Technology and Artificial Intelligent Systems in Business Development,” in *2022 5th International Conference on Contemporary Computing and Informatics (IC3I)*, Uttar Pradesh, India: IEEE, Dec. 2022, pp. 1602–1607. doi: 10.1109/IC3I56241.2022.10073410.
- [40] P. Lopes De Souza, W. Lopes De Souza, and L. Ferreira Pires, “ScrumOntoBDD: Agile software development based on scrum, ontologies and behaviour-driven development,” *J. Braz. Comput. Soc.*, vol. 27, no. 1, p. 10, Dec. 2021, doi: 10.1186/s13173-021-00114-w.
- [41] S. Amershi *et al.*, “Software Engineering for Machine Learning: A Case Study,” in *2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*, May 2019, pp. 291–300. doi: 10.1109/ICSE-SEIP.2019.00042.

References

[42] “Learn Time Series Tutorials.” Accessed: Jun. 17, 2024. [Online]. Available: <https://www.kaggle.com/learn/time-series>

[43] julien, “Top 3 easy sales forecasting methods for your retail business,” Ryax Technologies. Accessed: Jun. 14, 2024. [Online]. Available: <https://ryax.tech/prevision-ventes-facile-retail/>

[44] “What is Tkinter for Python?,” GeeksforGeeks. Accessed: Jun. 13, 2024. [Online]. Available: <https://www.geeksforgeeks.org/introduction-to-tkinter/>

[45] T. P. D. Team, “pandas: Powerful data structures for data analysis, time series, and statistics.” Accessed: Jun. 13, 2024. [OS Independent]. Available: <https://pandas.pydata.org>

[46] “What is Sklearn? | Domino Data Lab.” Accessed: Jun. 13, 2024. [Online]. Available: <https://domino.ai/data-science-dictionary/sklearn>

[47] Melanie, “Matplotlib: Master Data Visualization in Python,” Data Science Courses | DataScientest. Accessed: Jun. 13, 2024. [Online]. Available: <https://datascientest.com/en/matplotlib-master-data-visualization-in-python>

[48] “Introduction to NumPy.” Accessed: Jun. 13, 2024. [Online]. Available: https://www.w3schools.com/python/numpy/numpy_intro.asp

[49] “What Is Kaggle and What Is It Used For?,” Coursera. Accessed: Jun. 13, 2024. [Online]. Available: <https://www.coursera.org/articles/kaggle>

[50] “Store Sales - Time Series Forecasting.” Accessed: Jun. 14, 2024. [Online]. Available: <https://kaggle.com/competitions/store-sales-time-series-forecasting>

Abstract

This thesis proposes "ScrumAI," an adapted Scrum methodology for the development of intelligent systems. The work explores the characteristics of intelligent systems and the limitations of existing approaches. The core contribution is the redesign of the Scrum workflow by adding a new work cycle specifically for AI development, as well as new roles. The ScrumAI framework also emphasizes best practices to address issues like evolving specifications and the iterative nature of model training. A real-world case study performed on the development of a sales forecasting system demonstrates the practical application and benefits of the ScrumAI approach. We aim to bridge the gap between agile development and the needs of intelligent systems, providing a structured yet flexible framework to enhance the efficiency and success of AI projects.

Keywords: Intelligent systems, Agile development, AI system development, Software engineering, Scrum, Sales forecasting.

Résumé

Ce mémoire propose "ScrumAI", une méthodologie Scrum adaptée au développement de systèmes intelligents. Le travail explore les caractéristiques des systèmes intelligents et les limites des approches existantes. La contribution principale réside dans la refonte du flux de travail Scrum en ajoutant un nouveau cycle de travail spécifiquement dédié au développement de l'IA, ainsi que de nouveaux rôles. Cette méthodologie met également l'accent sur les meilleures pratiques pour aborder des problématiques telles que l'évolution des spécifications et la nature itérative de l'entraînement des modèles. Une étude de cas réalisée sur le développement d'un système de prévision des ventes démontre l'application pratique et les avantages de l'approche ScrumAI. L'objectif est de combler le fossé entre le développement agile et les besoins des systèmes intelligents, en fournissant un cadre structuré mais flexible pour améliorer l'efficacité et le succès des projets d'IA.

Mots-clés: systèmes intelligents, développement agile, développement de systèmes IA, génie logiciel, Scrum, prévision des ventes.