

POLITECNICO DI TORINO

Master's Degree in Computer Engineering



Master's Degree Thesis

**Web tool for quality control in an
automotive warehouse**

Supervisor

Prof. Renato FERRERO

Candidate

Riccardo CURELLO

APRIL 2025

Summary

This thesis focuses on the introduction and implementation of Agile methodologies within the software development team at FPT Industrial, specifically in the context of the Teardown Tool project. Agile, with a particular emphasis on Scrum, was integrated to improve development efficiency, communication, and adaptability in an environment traditionally unfamiliar with these practices.

The study begins with an overview of Agile methodologies, covering Scrum, Extreme Programming (XP), Lean Software Development, and Kanban, highlighting their relevance to the automotive software industry. Additionally, previous studies on Agile adoption in the automotive sector were analyzed to understand existing challenges and best practices. This provided a foundation for evaluating how Agile could be effectively introduced in a team unfamiliar with its principles. This is followed by an introduction to FPT Industrial, its business complexity, and the Digital Team, which played a key role in the project.

The core problem addressed in the thesis is the inefficiency in teardown analysis and warehouse management. Initially, teardown processes were largely manual, relying on spreadsheets and emails. The project aimed to replace these outdated methods with a structured, cloud-based system using AWS, Node.js, and React.js. Agile practices were adopted to facilitate iterative development, feedback loops, and cross-team collaboration. Additionally, the thesis proposes tailored solutions specifically designed to enhance the teardown process in the automotive sector, optimizing data collection, analysis, and reporting to improve efficiency and decision-making.

As Scrum Master, I introduced key Agile ceremonies such as Sprint Planning and Sprint Review, ensuring efficient sprint cycles. A major challenge was bridging communication gaps between technical and non-technical teams. My role was crucial in translating requirements across different disciplines, fostering collaboration, and even contributing technically when needed.

The implementation of Agile not only improved software development efficiency

but also opened the way for new business opportunities. The systematic tracking of teardown data supports circular economy initiatives by enabling the reuse of analyzed parts, reducing mechanical waste, and optimizing recycling efforts. Additionally, it allows for the collection of well-structured data that can be leveraged for improvement analyses by the Quality team. In the future, these insights could be used to develop tools that suggest specific analyses for teardown operators, helping them identify issues more effectively and streamline the diagnostic process.

Despite these achievements, challenges remained, including initial resistance to Agile adoption, workload management, and cross-team coordination. Future developments could focus on refining Agile integration more deeply, automating more aspects of teardown analysis, scaling the solution for other applications within the company, and customizing it for other regions.

This thesis demonstrates that Agile methodologies, when properly implemented, can enhance software development in the automotive sector, improving both technical efficiency and business outcomes. Moreover, having a strong technical background in addition to Agile expertise facilitates better communication between teams, enabling more effective problem-solving and decision-making.

Keywords: Project Management, Quality Control, Customer Service, Web Application, Cloud Architecture, Agile Methodology, SCRUM, FPT Industrial.

*To my grandparents,
my dear Teresa and Elio,
whose love, wisdom and strength
have always been a source of
inspiration in my life.*

Table of Contents

List of Tables	VIII
List of Figures	IX
1 Introduction	1
1.1 Agile Method	1
1.1.1 Extreme Programming (XP) Methodology	3
1.1.2 Lean Software Development	5
1.1.3 Kanban Methodology	8
1.1.4 SCRUM	10
1.2 The host company FPT Industrial	13
1.2.1 History and foundation	13
1.2.2 Business Complexity	13
1.2.3 External Suppliers and Partners	14
1.2.4 Digital Team	14
1.2.5 International Awards and Recognitions	15
2 Problem Analysis and State-of-the-Art Solutions	17
2.1 The Problem Context	17
2.1.1 Technical Center	18
2.1.2 Teardown	19
2.1.3 Problem Definition and Thesis Objectives	20
2.1.4 Overview of the work conducted so far	21
2.1.5 Objectives and expected impact of the project	22
2.2 Related work	22
2.2.1 Case Study: Using Agile Methodologies in Automotive Software Development	23
2.2.2 Case Study: Improving Warehouse Operations in Cargo Companies through Agile-based Warehouse Management System	24
2.2.3 Case Study: Agile Data Warehousing Project Management: Business Intelligence Systems Using Scrum	25

2.2.4	Case Study: Agile development in automotive software development: Challenges and opportunities.	27
2.2.5	Case Study: Problem reports and team maturity in agile automotive software development	28
3	Implementation and development in FPT	29
3.1	Context and Role in the Company	29
3.2	Project Description	30
3.2.1	Project technologies, tools, and methodologies used	31
3.3	Development and Implementation	35
3.3.1	Development Process	35
3.3.2	Development Technicalities	40
3.3.3	Challenges Faced	45
4	Conclusions	52
4.1	Summary of the Work	52
4.2	Key Achievements and Results	54
4.3	Limitations of the Project	55
4.4	Future Developments	56
4.5	Final Thoughts	56
	Bibliography	58

List of Tables

2.1	Relative Performance of Waterfall and Agile Projects When Run Side By Side	27
3.1	List of available user types in the Teardown Tool	40

List of Figures

1.1	Iterative Flowchart	4
1.2	XP Project Workflow	6
1.3	Kanban board elements	8
1.4	Scrum Process	13
2.1	Turin Technical Center	19
3.1	FPT Teardown Architecture (first version)	31
3.2	DevOps Teardown Project	33
3.3	FPT Teardown Business Process for Turin Warehouse	41
3.4	Menu and Instructions provided in the first sheet of the Excel . . .	42
3.5	TRR definition with items selectable from a list in Excel for bulk import	42
3.6	Item Details in Web Tool, general information and status	43
3.7	Web tool: Warehouse positions management	44
3.8	Turin Teardown Warehouse getting mapped	45
3.9	FPT Teardown Architecture (last version)	48
3.10	AWS Cloudfront WAF Allowed Requests	49

Chapter 1

Introduction

Thanks to the active collaboration with "*Politecnico di Torino*" and FPT Industrial, I have the opportunity to develop this thesis by presenting a case study on applying an Agile methodology in a complex project. This collaboration allows me to put in place and analyze the challenges and benefits of Agile practices within a real-world industrial context, providing valuable insights into their effectiveness in managing complexity, improving team efficiency, and fostering innovation.

The first chapter will explain the Agile methodology, why it is worth using it in software development, and the child methodologies that have derived from it. The host company, FPT Industrial, will then be introduced to provide an overview of what it does, its complexity, and its commitment to a global market.

1.1 Agile Method

In today's fast-paced and ever-evolving technological landscape, traditional project management methodologies often struggle to keep up with the demands of modern software development. In response to these challenges, Agile methodologies have emerged as one of the best approaches to managing projects efficiently, particularly in the field of software engineering, the main field it was created.

Since the publication of the Manifesto for Agile Software Development in 2001, known also as "*Agile Manifesto*", twelve principles were written by seventeen expert software engineers. They shared a common frustration against companies that were only focused on planning and documentation of workflows, losing the main goal: customer satisfaction. Not by chance, the first precept reported is "*the satisfaction of the customer and delivery of a valuable product*". [1]

Developers started thinking that software development lacked a clearly defined collaborative methodology, and Agile flourished for two reasons. "*First, they match*

the business need to deal with relentless speed and change, and second, they forge the workforce culture of the future." [2]

Another important topic is the company's competition beyond the simplistic view of software development. *"Agile organizations create chaos for their competitors, first by creating changes so fast that competitors are left gasping for breath; and second by responding quickly to competitors' attempts to change the market".*[2] Our society is based on a technological environment, and every business has its foundation on technology too. Every production process, service, or other business uses machines and software to produce and deliver products and earn from them. The technological park needs research and upgrades to be more productive and keep pace with the times.

Indeed, Agile guarantees flexibility, collaboration, and customer satisfaction, distinguishing itself from traditional waterfall approaches that rely on rigid planning and sequential development. Additionally, a rigid, pre-designed framework is not an effective solution in a software development workflow. This is because a high-level overview of the solution is not enough and does not address the consequences of development, cannot take into account problems arising later, lack of time, bugs, and other variables. Moreover, the estimate of time and cost may be wrong and could have an ever-ending development time, without any working product in the meantime.

By embracing iterative development, continuous feedback, and adaptive planning, Agile enhances team productivity and ensures that products meet evolving customer needs. In a real-world situation, things change a lot of times and very frequently, so it is obvious that requests also evolve very quickly and require more attention. Agile anticipates the need to allocate time for discussions and the reallocation of effort, needs, and priorities.

To achieve this result, the Agile methodology suggests splitting the work into short periods that correspond to the smallest amount of time needed to produce some tangible changes. This will to avoid too many functionalities in our software and the possibility of checking a working frame of the software. That's because the development of software also brings several bugs, and discovering them in a short period is better than carrying them for a long time. The more time passes, the harder it becomes to fix issues quickly because the developer has to study the code, functionalities, and strategies involved in their context again. The accumulation of bugs in later stages would increase the cost to test, discover, fix, and deliver. A late discovery of several bugs has significantly increased the cost and time needed. [3]

Delivery periods preferred are meant for Agile to be from two weeks up to two

months, but the less the better. During this period, team members are required to work on the project, check requirements, produce code, verify requirements reached, think about possible problems, discuss them, and propose improvements.

A delivered code doesn't need to be distributed directly to users. A chain of approvals with different persons and roles may be needed. It could cause slowness in publishing a new release because it requires some time for every block of the chain to read, understand, verify, and approve a small piece of new code. So, it is important to hug the *Continuous Delivery Approach*, which notes that developers should always keep a releasable version available and it should be updated after some delivery periods to keep it fresh, but also sufficiently tested.

A project is not meant to be completed alone; it requires a team that collaborates and maintains active communication to achieve its goals. As reported in one principle of the Agile Manifesto, *Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.* [1] Nevertheless, the team is not meant to be composed only of software developers, but also sponsors and users are an integral part of the process and should collaborate in the development.

The Agile Manifesto has been a milestone for developing, and sooner were soon born other methodologies inspired or that share some principles in common. Extreme Programming, Lean Software Development, Kanban, and SCRUM are examples of successful methods and frameworks used in software development even nowadays.

1.1.1 Extreme Programming (XP) Methodology

Extreme Programming (XP) is an Agile methodology that highlights the quality of the software and its flexibility in development. The first XP project was started on March 6, 1996, by Kent Beck as a response to the difficulties encountered in traditional models for software development. *"Of the four project development variables - scope, cost, time, and quality - quality isn't a free variable. The only possible values are 'excellent' and 'insanely excellent', depending on whether lives are at stake."* Beck affirms.

Customer and managers usually focus on the other three variables (scope, cost, and time) and developers should take into account the fourth one - quality. But it is not possible to have valuable software under stressful conditions of time and human resources (costs). The gold rule is that you can control three variables at a time, and the fourth will come as a consequence. *"The solution is to make the four variables visible. If everyone - programmers, customers, and managers - can see*

all four variables, they can consciously choose which variable to control. [4]

I can also deduce that the variables have strict constraints on each other. For example, you cannot increase a team in one shot just by giving more money to the project, but you have to increase by time the number of developers and teams. Or the quality aspect, if taken seriously over time, will soon create a better code and allow developers to program in a faster way, delivering a working code in less time.

Everything starts with planning, even in Agile projects. The main difference is that the plan is not strict and should not follow a step order. The first phase to do is to ask the customer a set of user stories. A user story is a tale that describes an action that should be performed. It could follow the rule of WHO, WHAT, and WHY (e.g. *"As a user I want to log in to my account so that I can start to work"*).

User stories are not like a huge file with all requirements, but they are short, simple, and at a high level. No technicalities are involved, and usually the customer uses his terminology with which he is familiar. About 80 user stories, plus or minus 20, is a perfect number to start a project and define a release plan.

XP fits very well in projects that require fast iterations and continuous adjustments to changes. Indeed, XP follows the Agile principle of creating small iterations that could deliver a working version with some additions to the code.

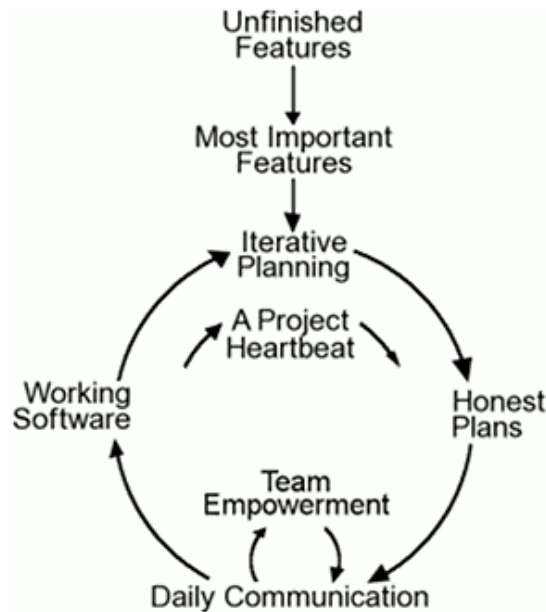


Figure 1.1: Iterative Flowchart

As reported in figure 1.1, at each iteration are grabbed the most important features described by user stories are grabbed to be added to our current state and

create a sustainable planning.

It also could include an open dialogue between customers, managers, and developers about the four variables discussed before. With constant (even daily) communication, it is possible to clear doubts about requirements and integrate different teams for a final working software solution.

It is important that at each cycle it is asked to retrieve the most important feature to develop is retrieved, since new user stories can arise and priorities could change for the company. In addition, focusing on a single feature lets developers work in a safe context and without distractions for a smarter solution.

Another important phase is the testing one, which is not only within the developers' competence but also involves customers. For each user story written by the customer, which was the starting point of a feature or a later modification, it is required to write down some tests with an expected result. Tests are previously done by developers while coding, but also with a stable version of the software.

Later on, it is asked of customers to perform their tests, which could pass or not. In the second case, it is important to write down a new user story describing the correct behavior it should have, emphasizing the misunderstanding.

After Acceptance Tests, a small release is done. This is useful for customers to get familiar with the software and use it for finding bugs, new user stories or new implementation ideas.

Even during the developing phase, you have access to a stable and working piece of the final solution. For the customer is a great involvement, keeps active, and gratifies the team.

The project evolution is made by growing steps and complexity with a strong base, tested several times, and a strong flexibility, which allows it to grow in functionalities.

An XP project flow can be summarized by the scheme reported in 1.2, which does not describe a linear behavior but focuses on the quality of the code and an iterative approach.

1.1.2 Lean Software Development

A different kind of approach could be analyzed with Lean Software Development. This term first appeared in 2003 when Mary and Tom Poppendieck published their book *"Lean Software Development: An Agile Toolkit"*, which outlined seven principles. Their work was heavily inspired by the *Toyota Production System (TPS)*, a manufacturing methodology developed by Toyota to optimize efficiency, minimize waste, and continuously improve processes [5].

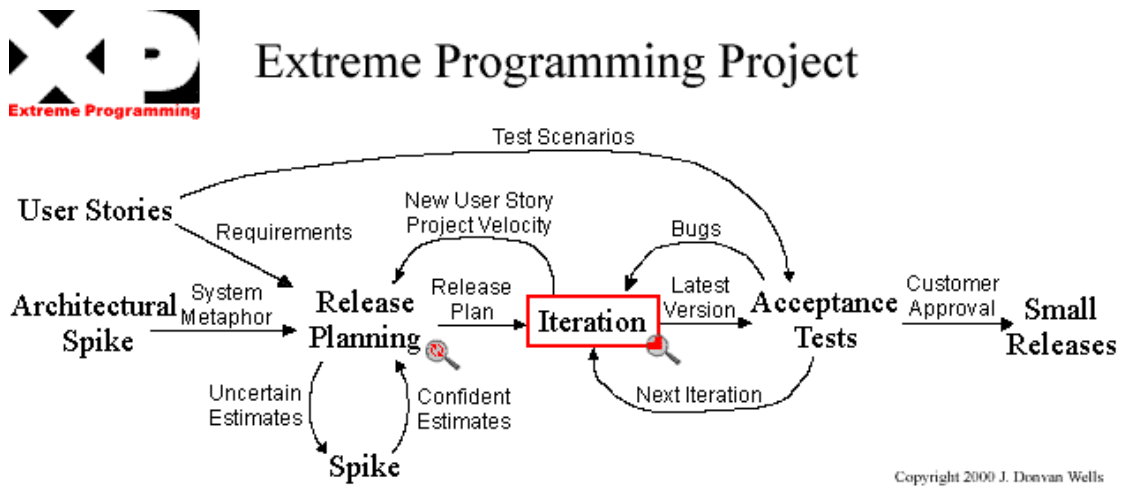


Figure 1.2: XP Project Workflow

The first principle is the one that also gives the name to the methodology, **Eliminate Waste**. Everything that is not necessary should be eliminated. It could be a piece of code, processes, real-world operations, bureaucracy, and so forth. The reason why some companies work better in comparison with the same business is that they focus on simplicity.

"The ideal is to find out what a customer wants, and then make or develop it and deliver exactly what they want, virtually immediately. Whatever gets in the way of rapidly satisfying a customer need is waste." [6]

Follows **Amplify Learning**, which is a key principle that emphasizes continuous improvement through experimentation, feedback, and adaptation. Unlike traditional development approaches that rely on extensive upfront planning, Lean encourages iterative cycles where teams can quickly test assumptions, gather insights, and refine their solutions.

This approach reduces the risk of costly mistakes by enabling developers to learn from real-world usage and stakeholder feedback. In the end, amplifying learning helps organizations stay competitive in a fast-changing market by embracing adaptability and continuous improvement.

Against the current, **Decide as late as possible** to avoid blocking creativity and smart solutions. Giving an ironclad decision limits the team from finding other solutions, and it may not even fit with future integration of code or business model. In this case, waiting until the problem arises could be useful and avoid the loss of time modifying the project later.

Having all pieces up to that point is useful instead of having a plan carved in stone. It is possible to find concrete studies in automotive companies that confirm that. *"While in most cases CE seeks to freeze specifications quickly, Toyota's engineers and managers try to delay decisions and provide their suppliers with hard specifications very late in the process. And, while conventional concurrent engineering reduces the number of prototypes, Toyota's suppliers seem to multiply prototypes, in some cases to an absurd degree."* [7]

Commonly to Agile, **Deliver as fast as possible** because it helps the team to be on the ball and close the cycle composed of design, implement, feedback, and improve. *"We're familiar with the refrain: 'Sure I can get it ready sooner, but it's going to cost you a fortune'. We suggest that 'faster is dearer' will now join 'quality costs more', an idea debunked."* [8] A faster delivery assures that customers get what they need now and improves satisfaction.

Empower the team is fundamental since end-users know what they need. Close collaboration between developers and end-users ensures that the software aligns with actual user requirements, rather than assumptions or technical specifications alone. With the help of a leader, is it possible to collect and translate requests into specifications useful to developers.

After all, developers could not imagine the real use of their software since it is not their job. Scheduling meetings, frequent integrations, testing, and visible charts could improve the final product refinement at regular intervals.

For building a valuable product, you need **Build integrity in**. *"The economic importance of product integrity overall cannot be underestimated."* [9] Software should be intuitive and do what the user expects. Conceptual integrity means that the system's central concepts work together as a smooth, cohesive whole, and it is a critical factor in creating perceived integrity.

"In short, conceptual integrity of the product not only makes it easier to use, it also makes it easier to build and less subject to bugs." [10] Integrity also includes the ability to maintain, adapt, and extend the software.

See the whole is the last principle that is the most to be achieved in large companies. Indeed, it is a fact that different teams over specialized and work independently. Each team tries to improve the work of its competence in an isolated way without taking into consideration the whole project.

Instead, this approach helps avoid optimized subsystems that do not improve the overall value of the product. It also promotes a systemic vision, where each decision takes into account the long-term impact, avoiding local solutions that could compromise overall efficiency.

1.1.3 Kanban Methodology

The Kanban method is a visual and iterative approach to work management, originally developed in the context of Japanese industrial production and later adapted for software development and business process management. Taiichi Ohno, the inventor of Kanban and an engineer at Toyota, thought that visualizing tasks could improve manufacturing efficiency and concurrency. The idea is based on studying inventory accumulation points on the production line to identify those inefficiencies and remove them.

Kanban is a Japanese word that means "signal card" in English. It summarizes the aim of this method. *"A number of kanban (or cards) equivalent to the (agreed) capacity of a system are placed in circulation. One card attaches to one piece of work. Each card acts as a signaling mechanism. A new piece of work can be started only when a card is available. This free card is attached to a piece of work and follows it as it flows through the system. When there are no more free cards, no additional work can be started. Any new work must wait in a queue until a card becomes available. When some work is completed, its card is detached and recycled. With a card now free, a new piece of work in the queuing can be started."* [11]

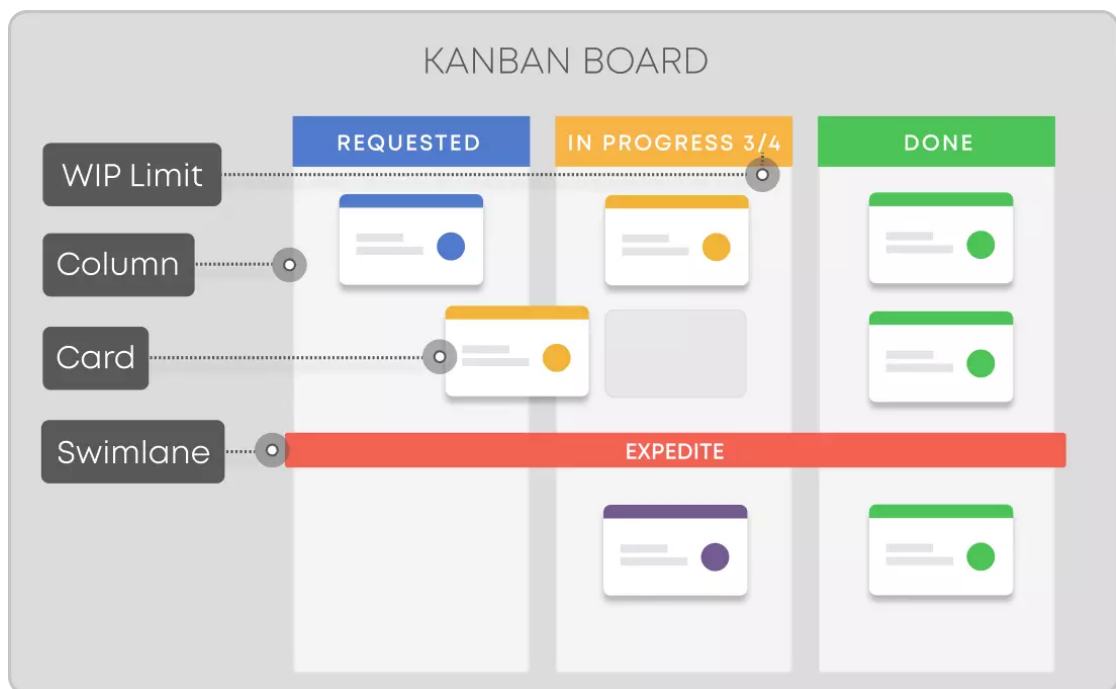


Figure 1.3: Kanban board elements

Each card could be marked with different colors to match priority or theme of interest, and put on a board. A classical board is divided into columns, many necessary corresponding to the steps of the process. At the beginning, simplicity is encouraged, so start with a few columns. New columns will be created later on, when the project evolves in complexity.

Now the team should agree on the maximum capacity of the columns; the number of cards cannot exceed this limit for each column that corresponds to the maximum Work In Progress (WIP).

The last step is to write down as many cards as you wish that are in the state on the most left column (e.g. *"To Do"* column). Next column, for example, is *"In Progress"* and can accept some cards that are moved from the left column, always considering the maximum limit.

Board and cards make it easy to discover where a possible project bottleneck is located. The division into columns lets people work on steps and encourages higher quality and greater performance. Every card is tracked on its change of state until it blocks or is completed.

"Kanban facilitates the emergence of a highly collaborative, high-trust, highly empowered, continuously improving organization. Kanban has been shown to improve customer satisfaction through regular, dependable, high-quality releases of valuable software." [11] The limitation on WIP highlights work overload and focuses on priorities.

With Kanban you can get metrics out like Lead Time and Cycle Time. The first one counts the time required to complete a card from start to end point; instead, the second one measures the time needed to complete one phase of the process. These KPIs can be useful to managers to understand phases that could be improved after a study of the results.

Another principle believe in a common language and rules between members of the team. For example, a card cannot be moved from "In progress" to "Done" without staging on the "Testing" column. Even the *Definition of "Done"* is important because it establishes when a card ends its cycle life. By the way, a common language avoids misunderstanding in the team and loss of card priorities.

Continuous improvement in Kanban is supported by regular feedback loops, such as:

- Daily Standup Meetings, short daily meetings to update on the work in progress.

- Flow Review, periodic analysis of the process to find opportunities for improvement.
- Kanban Retrospective, sessions dedicated to reflecting on what is working and what needs to improve.

Feedback is useful to tune and facilitate the progression of cards up to a state of completeness. Additionally, it stimulates work groups and creativity.

Since some metrics are collected, it would be foolish not to use them for improvements. The Kaizen principle is applied. The Kaizen principle, derived from the Japanese words kai (change) and zen (good), represents a philosophy of continuous incremental improvement. It emphasizes small, ongoing positive changes that collectively lead to significant enhancements in efficiency, quality, and performance over time.

Kanban is used in different environments bringing a significant benefit. It is possible to find validation in many studies like the one reported by an office team: *"Our early experience showed that Kanban helped the team get an overview of what was going on at any point in time. This allowed the team to move from working as individuals pulling tickets off queues to coming together as a team."* [12] Even if the team is limited to 2-3 people, the effort in creating a tidy board is worth it: complexity will come and work can continue without blockages.

1.1.4 SCRUM

Scrum is an Agile framework designed to facilitate the development of complex products through iterative and incremental progress. Originally created by Ken Schwaber and Jeff Sutherland in the 1990s, Scrum provides a structured yet flexible approach to project management, emphasizing collaboration, adaptability, and continuous improvement. The word "Scrum" refers to the game of rugby, where players fight all together to get the ball and win. Similarly, the team should work together to complete the goal. It is based on three key pillars: transparency, inspection, and adaptation, ensuring that teams can quickly respond to changing requirements and deliver high-value products efficiently.

- *Transparency* partially resumes what I already mentioned in Kanban on section 1.1.3, a common language and definition of done are the foundations for good communication and a healthy relationship in the team. In addition to that, all actors involved in the project should have visibility on the current work. This transparency also finds utility when the other two pillars.

- *Inspection* is in charge of investigating the current status of the project and the clarity of the requirements and their compliance. We've already seen in 1.1 that finding errors earlier is an advantage and benefit for the company in terms of costs and time. *"Every little while, stop doing what you're doing, review what you've done, and see if it's still what you should be doing and how you might do it better. It's a simple idea, but executing it requires thought, introspection, honesty, and discipline."* [13]
- *Adaptation* occurs when a person realizes that requirements have not been met and that the final project could be rejected. Therefore, applying changes during the development phase and remaining flexible to possible changes in requests. It turns out that the more people in the process have access to the developments, the more likely it is that discrepancies from the requirements will be found in time.

The team is composed of main figures suggested by the SCRUM framework which have a specific function and are meant to be an active part of the process. A good starting point is to have a clear understanding of each person's rights and duties so as not to override roles. Well-defined roles are Product Owner, Scrum Master, and Development Team.

"The Product Owner (PO) is the empowered central point of product leadership. He is the single authority responsible for deciding which features and functionality to build and the order in which to build them." [14] Takes the buyer's side and is the point of reference for the project. The PO has clear ideas about the features that the product must have and writes user stories to be put in the backlog.

Scrum Master (SM) is the leader of the group (not a manager). His role is to make the development teams interact and coordinate the evolutionary process of the project. Furthermore, he intervenes to help the development team in case of blocks that prevent development by solving them. It's a facilitator: it is his responsibility to oversee the overall Scrum process and adoption over the Development Team. For that reason, he works closely with the PO.

Development Team is usually composed of five to nine persons who develop features described in user stories. Having more people involved does not mean that they lead to greater development. In this case, more development teams with parallel Scrum logic are needed. Usually they are architects, programmers, testers, database administrators, UI designers, and so on.

Scrum organizes work into *Sprints*, time-boxed iterations (typically 1–4 weeks), during which cross-functional teams collaborate to produce a potentially shippable

product increment. Sprints have specific tasks to do that are collected in *Sprint Backlog*.

Sprint Planning is when it is chosen what to include in Sprints. During this event, the Product Owner, Scrum Master, and Development Team collaborate to define the *Sprint Goal* and select the highest-priority items from the *Product Backlog* to include in the Sprint Backlog. The team breaks down these items into smaller tasks and estimates the effort required. The key objective is to establish a clear, achievable plan for the upcoming Sprint, ensuring alignment between business objectives and development work.

Differently, *Daily Scrum* is a short, time-boxed meeting (usually 15 minutes) held every day of the Sprint. The Development Team members synchronize their activities, discuss progress, and identify any potential impediments. Each team member typically answers three key questions:

- What did I complete yesterday?
- What will I work on today?
- Are there any obstacles blocking my progress?

The purpose of the Daily Scrum is to foster collaboration, enhance transparency, and enable quick decision-making, helping the team stay focused and adapt to emerging challenges.

At the end of the Sprint, it is organized a *Sprint Review* is organized where the Development Team demonstrates improvements and completed work. During this ceremony, stakeholders directly or indirectly mark features as "Done" or propose modifications to be added to the Product Backlog with a linked user story. This event promotes adaptability and ensures that the product is evolving in alignment with business needs and user expectations.

The *Sprint Retrospective* is the final Scrum event, aimed at continuous improvement. *"The Scrum Team inspects how the last Sprint went with regards to individuals, interactions, processes, tools, and their Definition of Done. Inspected elements often vary with the domain of work. Assumptions that led them astray are identified and their origins explored. The Scrum Team discusses what went well during the Sprint, what problems it encountered, and how those problems were (or were not) solved."* [15]

When a Sprint is completed, *Increment* will also include all elements of the past Sprint to have a well-documented story to be reviewed if, in the future, it is asked to remember how a process works or linked changes.

The overall Scrum process could be summarized and visualized in Figure 1.4.

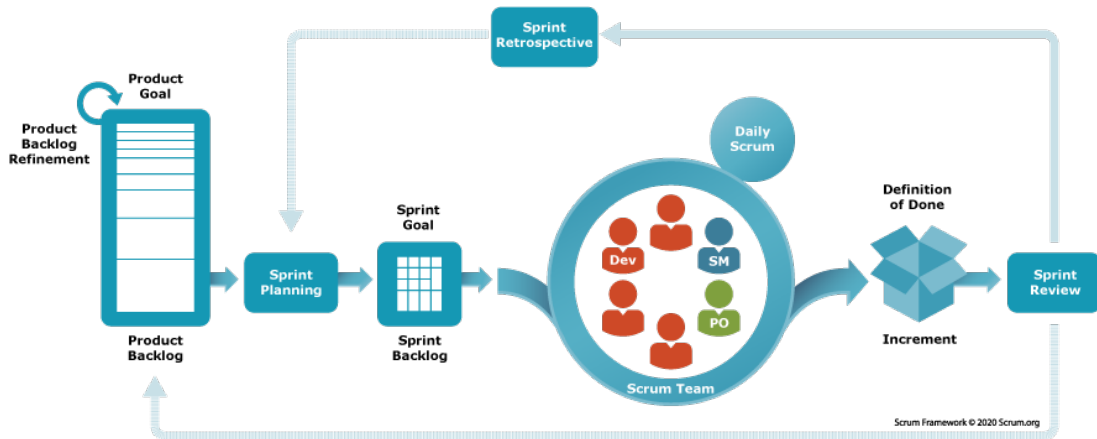


Figure 1.4: Scrum Process

1.2 The host company FPT Industrial

1.2.1 History and foundation

FPT Industrial is a global leader in the design, production, and sale of powertrains for on-road, off-road, marine, and power generation applications. Established in 2005 as a division of the Fiat Group, the company has since evolved into a key part of the Iveco Group, specializing in the development of innovative and sustainable propulsion solutions. With a strong focus on research and development, FPT Industrial has played a crucial role in advancing technologies such as Common Rail fuel injection systems and alternative powertrains, including natural gas and electrification.

1.2.2 Business Complexity

The company employs approximately 20,500 people worldwide and operates a vast industrial network with 10 manufacturing plants and 7 research and development centers spread across Europe, North and South America, and Asia.

This extensive footprint allows FPT Industrial to serve a diverse range of markets, providing advanced powertrain solutions to automotive manufacturers, agricultural and construction equipment producers, marine companies, and industrial power

generation businesses. Through its global presence and commitment to innovation, FPT Industrial continues to shape the future of sustainable mobility and high-performance propulsion systems.

1.2.3 External Suppliers and Partners

FPT Industrial operates within a highly complex and extensive supply chain, requiring a robust network of suppliers and external partners to meet the demands of its large-scale production. Given the global reach of its operations, the company relies on a strategically distributed supply base to ensure the continuous availability of high-quality components and raw materials.

The manufacturing process demands strict logistical coordination, as parts and assemblies must be sourced from different continents and delivered just in time to production plants. This requires advanced logistics management, integrating predictive analytics, real-time tracking, and automated warehousing to optimize efficiency and reduce lead times.

To maintain excellence in product quality and reliability, FPT Industrial enforces stringent supplier standards, ensuring compliance with industry regulations, sustainability criteria, and performance benchmarks. Suppliers must adhere to rigorous certification processes and continuous improvement programs to align with the company's commitment to innovation, durability, and environmental responsibility.

On the technological front, FPT Industrial adopts a hybrid approach to digital transformation, leveraging both internal development teams and external commissions to maintain a strong and agile digital infrastructure. This ensures seamless integration across enterprise resource planning (ERP) systems, manufacturing execution systems (MES), and supply chain management platforms.

The company continuously invests in data-driven decision-making, IoT connectivity, and cloud-based solutions to enhance operational efficiency and strengthen collaboration with suppliers and partners worldwide. Through this approach, FPT Industrial reinforces its technological leadership while maintaining the agility necessary to adapt to the evolving demands of the global market.

1.2.4 Digital Team

My project has been done in the *Digital Team*, which is part of Customer Service and is tasked with developing and deploying innovative software solutions that contribute to enhancing the service provided to customers. The team is important in assisting FPT in digitalizing its after-sales service strategy, making it more

efficient, agile, and data-driven.

At the core of the Digital Team's operations is the *Control Room*, the hub for telematics data monitoring and analysis. With real-time connectivity of vehicles, the team can remotely diagnose issues, predict potential failures and provide proactive customer care. This digital support model maximizes fleet management by minimizing downtime to a bare minimum, scheduling maintenance to the optimum, and maximizing overall operational efficiency.

Beyond telematics, the Digital Team is tasked to develop other creative digital solutions for enriching the after-sales experience. Through leveraging the power of cloud computing, data analytics, and AI-picked insights, the team is striving to facilitate easy customer engagement, quality and on-time information delivery, and the service experience to be frictionless.

The constant shift in customer expectations in the automotive and industrial industries necessitates the transition towards digital transformation. The Digital Team, therefore, acts as an incubation center, connecting different departments and mapping solutions to FPT's long-term strategic goals.

1.2.5 International Awards and Recognitions

FPT Industrial has established itself as a global leader in powertrain innovation, earning numerous awards and international recognitions for its technological advancements, sustainability efforts, and overall excellence in the industry. These accolades highlight the company's commitment to innovation, efficiency, and environmental responsibility, reinforcing its position as a benchmark in the global powertrain sector.

One of the most notable recognitions comes from the Diesel of the Year Award, which FPT Industrial has won multiple times, last in 2020 [16]. It has previously won for its 3.2-liter F32 engine in 2008 and the Cursor 16 engine in 2014.

This prestigious prize, awarded by Diesel magazine, acknowledges groundbreaking engine developments that push the boundaries of performance and sustainability. The company has been recognized for its high-efficiency diesel engines and its pioneering work in alternative propulsion systems, including natural gas and hybrid technologies.

FPT Industrial has also been honored with various sustainability and environmental awards, particularly for its commitment to reducing CO₂ emissions and integrating circular economy principles into its production processes.

The company actively works on carbon-neutral solutions, such as biomethane and hydrogen-powered engines, earning accolades from environmental organizations and industry associations worldwide. [17] [18] [19]

In the digital and technological innovation sphere, FPT Industrial has been acknowledged for its advancements in smart manufacturing, AI-driven predictive maintenance, and IoT connectivity within its powertrain solutions.

These achievements have been celebrated in various industry conferences and technology summits, solidifying FPT Industrial's role as a leader in the future of mobility and industrial power solutions.

Through these international recognitions, FPT Industrial continues to demonstrate its leadership in engineering excellence, sustainability, and innovation, positioning itself as a driving force in shaping the future of powertrain technology.

Chapter 2

Problem Analysis and State-of-the-Art Solutions

2.1 The Problem Context

In today's business world, customer satisfaction has become the key to business success. Satisfied consumers have been found to buy the same product again and again and recommend the business by becoming informal agents or advocates, thereby growing the business in a natural, spontaneous, and powerful manner through word-of-mouth. This finding led to the development of relationship marketing to establish and maintain long-term customer relations based on empathy for individual buyers [20].

A crucial element of ensuring customer satisfaction is the quality of customer service. Effective customer service is not just resolving issues and answering questions; it also anticipates customer needs, creating a positive experience as a whole. Customer Relationship Management (CRM) software enables businesses to efficiently manage customer interactions, interpret feedback, and provide timely and individualized solutions [21].

Thus, in brief, customer satisfaction and excellent customer service in this very competitive era are of core concern to one another and are of prime importance in creating strong, lasting relationships. These, in turn, affect a company's reputation and profitability significantly.

2.1.1 Technical Center

FPT Industrial realizes that exceptional customer care is not just the simple provision of world-class engines and powertrain solutions. For the best customer experience, the company has embraced the principle that properly trained dealers and OEMs (Original Equipment Manufacturers) are key to post-sales care.

By equipping its partners with deep technical knowledge and practical skills through training, FPT Industrial ensures customers get immediate and effective assistance from the frontline. This approach not only enhances the credibility and longevity of FPT products but also ensures clients' confidence and long-term relationships.

FPT Technical Center is a cutting-edge technological center for engine and component diagnostics and training with the most advanced technologies. The center combines theoretical knowledge and practical experience, offering workshops with engine walkarounds, diagnostics sessions, and classroom training. Designed as a means of marrying theory and hands-on learning.

The center not only serves as a diagnostics and repair training facility but also as a direct field experience for students and distributors where they can disassemble, diagnose, and reassemble high-technology products.

This initiative has a global reach, with not only the Turin plant but also the Sete Lagoas (Brazil), York (USA), and Pune (India) plants participating (future centers will come). With the involvement of over 300 technical employees in advanced technical training programs and with the participation of over 28 countries, from Europe to the Middle East and Africa, the FPT Technical Center assumes the role of a global pole for training activities.

Being part of the Customer Service division, the center is equipped with:

- An FPT training center for theoretical and practical diagnostics, including in-person and distance learning courses.
- A disassembly area for engine analysis, supported by a dedicated digital platform.
- A diagnostic laboratory for high-level testing.
- A historical gallery, home to six legendary engines that have traced FPT Industrial's history.

These centers are testaments to FPT Industrial's commitment to providing high-quality technical training and promoting excellence in the fields of technology and engines.



Figure 2.1: Turin Technical Center

2.1.2 Teardown

One of these areas is dedicated to disassembly engine analysis, the Teardown Area. It is set up with dedicated teardown benches, remote and on-site diagnostics areas in order to provide a professional customer experience.

The Engine Disassembly Area includes sophisticated test beds for every category of products from light-duty (2–4 liters) through heavy-duty Cursor and V20 series to middle-range Nef series engines. There are specially designed workstations for electrified powertrain pieces, including high-voltage battery packs, power electronics, and ePowertrain systems.

A variety of complicated diagnostic procedures are carried out within this facility, including:

- Turbocharging system analysis
- Piston seal integrity tests
- Compression and valve timing tests
- Fuel injection system and aftertreatment tests

- Electronic control unit (ECU) fault code diagnosis

Technical experts also examine prior usage histories to determine whether the engine has experienced abnormal conditions such as excessive loading, overheating, poor quality fuels, or improper maintenance. To get these results, some tools are used and Operators are required to track and provide a report of each component.

Based on the detailed analysis conducted in the Engine Teardown Section, a conclusion is made on the future of the engine or the component. Based on the result, the unit may be repaired or refurbished, replaced under warranty, or, where the issue falls outside the warranty terms, the warranty claim is denied. Moreover, in certain cases where a replacement is not feasible or justified, a refund can be issued to the customer.

This systematic approach ensures fair and transparent solutions and maintains the highest quality and reliability standards of FPT Industrial products.

Additionally, the results of these analyses are utilized for continuous product development, allowing technical teams to identify recurring issues, implement design improvements, and streamline maintenance processes.

All data is collected to ensure traceability of official interventions carried out on engines and for research purposes. Indeed, they will be studied by the FPT Quality Team to improve the construction and performance of engines produced.

Working components can also be assembled to get a refurbished engine and/or sold individually as second-hand components. Reselling would reduce waste of working components, use of new materials and keep a green direction for the company.

2.1.3 Problem Definition and Thesis Objectives

FPT Industrial asked to develop a web tool (a web application) to be used for Teardown operations. It should follow and track all steps necessary from a dealer to the final destination of the engine/component taken in the exam.

Different kinds of users will log in to the application and should have different levels of visibility and permissions for return requests. Another set of visibility is based on different involved warehouses and regions.

A Return Request starts from a Dealer, OEM, or a Quality Operator (more commonly to happen) and could have one or more engines or components. After a defect description and some identification numbers required to be inserted in the tool, engine, and components are shipped in a box and arrive directly or indirectly, in case of urgency or not, at a FPT Warehouse. The Logistic Operator will track

the arrival of each component and place the box in the warehouse according to a free spot detected by the application.

The second part is about analysis and includes the Teardown Operator, who tests the engine or component. In case of an engine, probably the operator needs to disassemble the engine in children components and they are written down in the application for further analysis. When the cause of the problem has been discovered, the Teardown Operator will complete a survey report and attach some photos/videos to certify it.

At the end, the Warehouse Manager will compile a Final Report that will be read by the dealer/OEM and choose a Final Destination of the engine/component based on the Teardown Operator's report. The engine/component will be processed as Final Destination says.

2.1.4 Overview of the work conducted so far

The Teardown project started in April 2024 with the first Technical Center Area built in Turin. Teardown was necessary because it is the link between customer, quality, warranty, and recovery.

It starts with a low flow of engines and components coming into the area and to be analyzed, but soon it starts its grow. The tools used for the analysis were the basic ones: emails and Excel files. For the analysis conducted, a survey was built to track and build a report that collects data and could be useful to the quality team to understand what was wrong with the engine/component.

Since the project became large, customer care's digital division was involved in building a custom solution for this area. The idea to develop a web-based application came out only two months later.

A Product Owner was identified, and a first team was built. The project development was entrusted to an external company that should write the code to be run on the FPT digital environment, security, and authentication system.

After several months of discussions, a list of requirements was produced, and for each feature was already decree a business priority, considering evolutions. In September 2024, the architecture was built in the Amazon Web Services (AWS) environment after some meetings with the development team, the security team, and the infrastructure team. During the meetings, the discussion was very debated, and several architectural changes were proposed for the development of this project.

2.1.5 Objectives and expected impact of the project

As previously reported, the web app project was a strong need raised by the fact that the number of engines/components to be analyzed increased. Furthermore, it was understood that it could come to define a standard for the business and reduce the compensation costs in cases of warranties requested with problems attributable to customers.

The historicization of returns makes it possible to trace the history of the engines/components and to have a list of the interventions officially carried out by the manufacturer. Furthermore, it allows the collection of data that can be used for future analyses.

The analyses can be retrospective for the improvement of the construction of the engines (materials, construction, components, suppliers, ...) or return campaigns or predictive analysis of failure.

Thanks to the use of the management tool that was requested, it is expected to be possible to increase the load of analyses to be conducted, bringing the number to 500-700 engines/components analysed per year per region.

Considering that the project currently has four teardowns, one per region, it is possible to notice the volume of analyses, data, and costs that can save tens of thousands of euros per engine.

Indeed, the web app will also manage the warehouse and track available locations. This means that the warehouse will not receive items if there are no available positions, but a large part of the items will be available in a buffer warehouse, and, upon request of the manager, it will be possible to order the movement of these.

Lastly, the integration of digitalization, automation, and structured data analysis will provide FPT Industrial with a powerful tool for continuous improvement, further cementing its position as a sector leader in quality assurance and after-sales service excellence.

2.2 Related work

Different case studies have been analyzed to construct a complete image of the utilization of agile methodologies and teardown systems in the automobile industry.

The case studies illustrate how companies have reacted to the increasing complexity in the development of automobiles, employing agile frameworks and teardown diagnostic techniques to enhance product quality, reduce time-to-market, and overall boost efficiency. By learning from real-world implementations, issues, and

best practices, this analysis offers a useful glimpse into how organizations can make their processes efficient with maximum reliability and innovation.

2.2.1 Case Study: Using Agile Methodologies in Automotive Software Development

Savitha et al. applied agile methodologies in the automotive sector based on the sector's shift towards digitalization and the need for agile software development practices. [22]

The automotive industry is also facing a paradigm shift driven by trends such as autonomous driving, increased connectivity, electrification, and rapid digitalization.

All these necessitate the adoption of agile software development methodologies to foster accelerated development cycles and cope with increased complexity. The reviewed article provides details of how automotive companies are adopting agile practices into their software development lifecycle.

To address these issues, the paper discusses solutions such as scaling agile methodologies from software teams to encompass the entire multidisciplinary vehicle development process.

Such as establishing cross-functional teamwork, engaging customers more, and acquiring feedback throughout the development process. Further, combining agile development with software product line engineering has been studied as a solution for addressing complexity in automotive systems effectively.

The case study highlights the revolutionary impact of agile approaches in the automotive industry, capturing both the challenges and steps to successful adoption. Different Agile frameworks are put in comparison to choose which one is the best to adopt in different situations, with pros and cons..

"Scrum is more flexible, with the possibility of co-creation with excellent customer participation in iterations (sprints)" [22] in case of not non-large-scale project.

Furthermore, *"Security and Quality practices can help the organizations to integrate security measures as well as external quality activities into technologies that are used in production projects" [22], that could help in next steps since the tool requested should be placed under the lens of Company's Security Team.*

2.2.2 Case Study: Improving Warehouse Operations in Cargo Companies through Agile-based Warehouse Management System

Warehouse operations are a crucial part of the global logistics industry as they ensure effective management of stock, distribution, and order fulfillment. Traditional warehouse management systems (WMS) have drawbacks of inflexibility, the absence of real-time visibility, and inefficiency in dispatch and distribution. These create delays, stock discrepancies, and reduced customer satisfaction.

To address such issues, agile methods have been increasingly used in warehouse management, as evidenced by Sihombing [23]. This case study explains that the deployment of an agile-based warehouse management system (AWMS) can significantly enhance warehouse operations in the freight forwarding industry through efficiency, adaptability, and responsiveness to dynamic customer needs.

Traditional warehouse operations are exposed to several operational issues, including:

- Poor inventory management, which leads to delays in fulfilling orders.
- Inadequate real-time tracking, leading to inventory discrepancies.
- Rigid processes that struggle to adapt to unpredictable demand.
- Limited integration with Enterprise Resource Planning (ERP) systems, creating information flow gaps.
- These inefficiencies drive costs and negatively impact the quality of services, and therefore, cargo companies must resort to innovative measures like agile-based warehouse management.

The report highlights how agile practices optimize warehouse operations by:

1. **Enhancing Collaboration:** Agile fosters cross-functional cooperation among warehouse staff, logistics groups, and IT teams to quickly address operational bottlenecks.
2. **Improving Responsiveness:** Through iterative processes and continuous feedback loops, companies can refine warehouse workflows to fit changing demands.
3. **Enabling Real-Time Monitoring:** Agile-based systems integrate real-time monitoring systems, providing accurate inventory visibility.

4. Automating Order Processing: Agile WMS dispatching and inventory management have been automated, reducing human error and improving speed.
5. They enable cargo companies to optimize operations, reduce errors, and enhance service levels.

In his research, Sihombing followed a structured three-step approach to analyze and implement an agile-based warehouse management system (AWMS) in cargo companies, ensuring measurable improvements in operational efficiency.

User need analysis is the first step, in which key stakeholders are analyzed and, interacting directly with potential users, features and functionalities required are collected.

Next is the turn of *Application development*, that is, the most time and resource-consuming part in which the application is developed. In this phase a very heterogeneous team (or teams) works together with different roles, following Agile methodology.

Finally, *User acceptance evaluation*, when the application is ready to be tested by end-users. *This stage includes functional testing, usability evaluation, and beta trials with end users.* [23]

The implementation of agile principles in warehouse management represents a breakthrough in the freight forwarding industry. By allowing for more flexibility, visibility, and automation, an agile-based WMS allows companies to provide better service levels and remain competitive in the evolving logistics landscape.

This case study provides a compelling argument for the broader application of agile principles in warehouse management, emphasizing the necessity for continued innovation and digitalization in modern logistics.

2.2.3 Case Study: Agile Data Warehousing Project Management: Business Intelligence Systems Using Scrum

Ralph Hughes presents a case study of using agile practices within data warehousing projects. [24] The study explores how the inefficient data management and business intelligence (BI) systems of a large organization applied Scrum to enhance the delivery of projects, quality of data, and flexibility.

The firm in question was experiencing several issues with data warehousing activities. Data warehousing project development cycles were taking far too long. This had adverse impacts on decision-making processes since business leaders were

not receiving timely, accurate information. Further, the rigidity of traditional data management systems prevented flexibility to accommodate changing business needs, limiting the organization to remain competitive in a rapidly evolving marketplace.

With this knowledge, the company intended to embrace agile methods, specifically Scrum, to redefine the development of its data warehousing and BI system. The move towards an agile-centric approach in data warehousing was a drastic deviation from the traditional way of handling projects.

Ralph Hughes shows that "*dashboarding*" and "*data integration*" are the main parts of a warehouse. The first one displays data through graphs and tables; the organization and development are quite easy for a warehouse. The second one is more complicated to achieve since very often data is dirty and needs to pass an ETL process (Extract, Transform, Load). So, he suggests following some steps:

- extracting necessary data from source systems
- cleansing it of errors and standardizing its formatting
- integrating the cleansed records with data from other source systems
- transforming integrated records into dimensional structures for easy data exploration

Furthermore, data accuracy and reliability were significantly improved. Feedback loops inherent in the Scrum process enabled early detection and rectification of data inconsistencies, and as a result, business intelligence reports were based on cleaner and more reliable sets of data.

These results are possible thanks to teamwork and continuous feedback through Scrum ceremonies. An early detection of defects and errors in business processes or software development reduces the time and cost of fixing. It is also proposed a table (Table 2.1) of comparison between a waterfall and an agile project.

This case study illustrates how Scrum is used in data warehousing and can bring about transformational efficiency, quality of data, and responsiveness in business. By adopting agile philosophy, the firm was able to do away with the limitations of traditional project management, reducing development time, enhancing collaboration, and increasing flexibility.

The experience confirms the value of Scrum and iterative development outside software development, demonstrating how agile approaches can make complicated data-driven projects better and business intelligence systems dynamic, current, and customized to organizational requirements.

Project Aspects	Waterfall Project	Agile Project	Agile's Advantage
Project definition (person months)	15	4	3.8x
Target tables loaded	8	20	2.5x
Developer hours per table	1100	400	2.8x
Development expense per table	66K	26K	2.5x
Defects found - system test	Four dozen	10	>4x
Defects found - UAT	Four dozen	0	Infinite
Trouble tickets - first 9 months	Scores	0	Infinite

Table 2.1: Relative Performance of Waterfall and Agile Projects When Run Side By Side

2.2.4 Case Study: Agile development in automotive software development: Challenges and opportunities.

This paper provides an interesting discussion on the implementation of Agile methodologies in the automotive industry. [25] The study was conducted at Volvo Car Corporation to improve the software development process.

The research analyzes the transition from a traditional development approach to an Agile methodology, motivated by the increasing complexity of modern vehicles.

Through 29 semi-structured interviews, company documentation, and literature reviews (which are among the six possible sources of data: documentation, archival records, interviews, direct observations, participant observations, and physical artifacts [26]), the authors examined various aspects of Agile adoption. The key areas investigated include ‘requirements’, ‘roles and competencies’, ‘communication’, ‘process and phases’, ‘agile understanding’, ‘quality improvement’, and ‘co-location’.

The study found that there is a lack of a standardized process, an excessive

workload with a 1:3 ratio (one person managing three projects), and a perception of Agile as an additional burden rather than an improvement.

The focus tends to shift from the overall process to individual deliveries, leading to a fragmented vision of the project. However, the adoption of Agile promotes a knowledge-sharing culture and cross-functional collaboration, which enhances efficiency in handling different tasks by leveraging shared information.

Moreover, the strict separation of concerns does not improve knowledge-sharing and instead leads to a low level of domain understanding among team members.

2.2.5 Case Study: Problem reports and team maturity in agile automotive software development

The study conducted by Gren and Shepperd [27] focuses on the *"human factors"* influencing individuals in an Agile project.

This research, like the previous one, was also carried out in an automotive company—Volvo. The authors examined the implementation of Agile methodologies within projects and conducted 82 interviews to gather insights.

Their investigation was driven by prior work from Moe et al. [28], who *"concluded that one of the barriers to agile team autonomy was a lack of system support for teams and reduced external autonomy"*.

Another key reference was Hodgson and Briand [29], who identified *"a trend that team members without much previous agile knowledge prefer the latter while experienced agile practitioners prefer the former"*. [27].

The study analyzed the average time required to correct errors across four stages of team maturity, progressing from a newly formed group to a *Work and Productivity* team. The results revealed that *"when the teams are newly formed [...], the average age of issues is higher for those teams"* [27].

Ultimately, the findings highlight that it is normal for a high number of issues to emerge in the early stages of Agile adoption. However, as teams mature, fostering dialogue and collaboration—key principles of Agile methodology—leads to improved efficiency and faster issue resolution.

Chapter 3

Implementation and development in FPT

In this chapter, the activities carried out during my work at FPT Industrial are presented, with specific reference to my contribution to the project. The chapter provides a comprehensive description of the development process, with the different phases, methodologies adopted and key technical decisions taken throughout its progress.

The discussion will include an overview of the initial requirements and their development throughout the process to achieve the company's strategic goals. Further, the problems encountered while working on the project will be talked about, such as technical constraints, integration problems and process enhancements. Each problem will be discussed along with the proposed and applied solutions, emphasizing the application of agile methodologies and collaboration in resolving them.

3.1 Context and Role in the Company

Up to now, the Analysis Phase has been completed, and the environment has been built, so the development of the application could start. In the last days of October 2024, I got into the project thanks to the thesis reporting from "*Politecnico di Torino*".

I achieved from my tutor an important role in the project: Scrum Master. As seen in Section 1.1.4, SM is the right arm of the PO and helps him to coordinate the work in different teams. The SM plays a really important role within the team because he is in charge of being the facilitator and has to cover a hybrid position

between the teams.

At first, it wasn't easy to understand what to do. Having been immersed in a large reality with different actors and roles, it was necessary to settle in and get to know the teams and the project.

The first step was to read all the files available with the requirements and highlight doubts and inconsistencies that emerged. Since I was like a blank canvas, it was an easy task because I was not influenced by preconceptions or memories related to past conversations or thought constructs.

Secondly, I got to know teams and people working around the project (and not only) to get familiar with colleagues and have reference points for the various fields. Alignment calls were addressed with the development team's project manager to clear doubts previously marked.

Since I have a minimum of experience in the realization of projects, and I have always been interested in the realization of a product, having a 360-degree vision of the project, I also compared myself with the team that will use the tool. Due to time and location issues, others couldn't cover this fundamental role of dialogue with those who currently work in the field.

Another important task that was asked of me was to understand the steps required to go through the process of going to production. Many puzzle pieces are needed to go on with the process, and each piece is provided by a team in this complex company. Every time, I had to discover which team is responsible for performing a certain action or issuing permissions to a resource. I say discover because there's no written procedure.

Last but not least role is the programmer one. It wasn't a request directly from FPT, but many times it was a necessary skill to go faster in some operations or get out of time-consuming troubles. I'll present some examples in Section 3.3.3 Challenges Faced.

3.2 Project Description

Meeting the off-the-shelf requirements outlined in Section 2.1.3 is a complex task. However, the process of achieving these results is far from trivial.

Translating high-level requirements into an effective and functional implementation involves several steps, including careful planning, iterative development, and continuous validation.

One of the main challenges is to make sure that the solution conceived is aligned with business needs as well as technical feasibility. Requirements evolve over the project duration due to changing priorities, integration concerns, or unforeseen constraints, and therefore, agility is crucial. Additionally, the complexity of system architecture, security limitations, and performance optimization must be addressed in a systematic approach to prevent bottlenecks or inefficiency.

3.2.1 Project technologies, tools, and methodologies used

For this project it was used an AWS solution was used to get a fast, scalable, and distributed architecture, since it is useful that a lot of people would connect to the application from different regions.

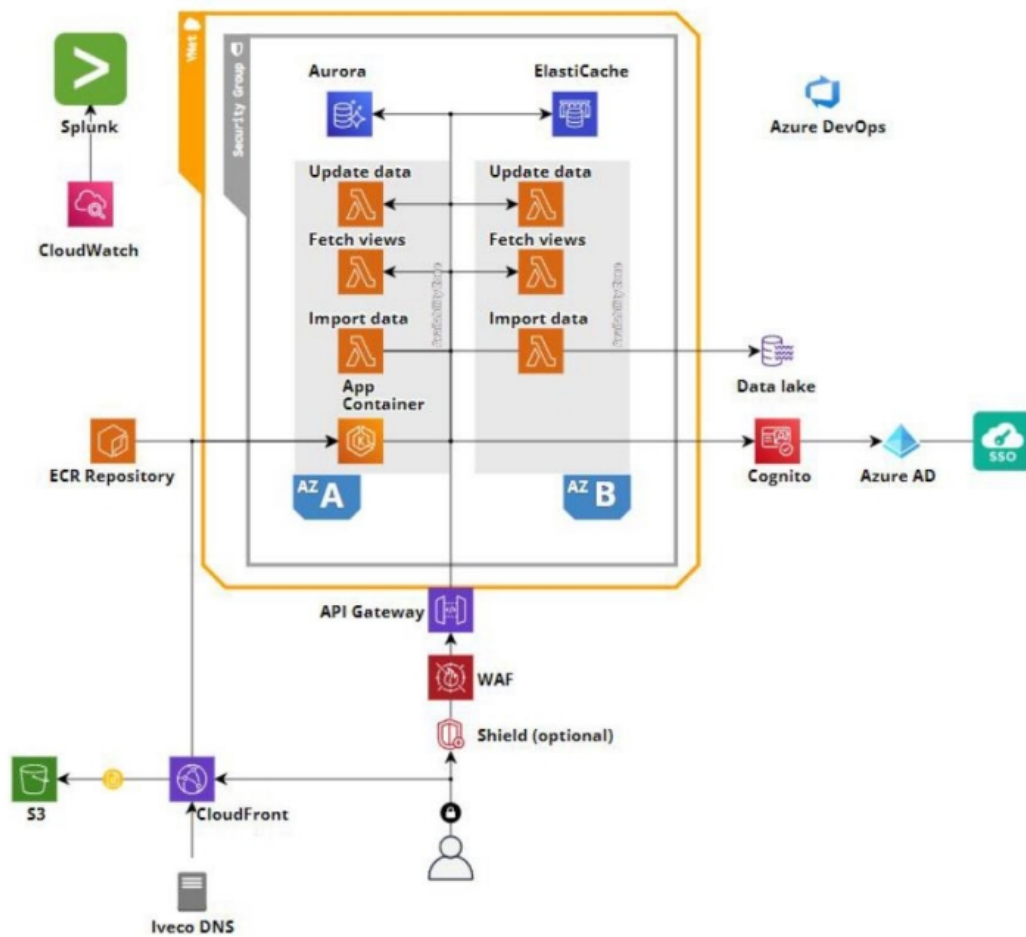


Figure 3.1: FPT Teardown Architecture (first version)

The architecture designed by the infrastructure team is presented in Figure 3.1 and was the outcome of numerous meetings and planning sessions. It is a composite solution with several resources, and security and scalability are maintained by Virtual Private Clouds (VPCs) and Web Application Firewalls (WAFs).

To give a clear understanding of the system's main building blocks, I describe the main resources with a key influence on the development of the project.

- **CloudFront:** Amazon CloudFront was implemented as a global Content Delivery Network (CDN) to distribute content securely with low latency. It ensures fast access to the application for users across different regions, improving performance and reducing load times.
- **EKS (Elastic Kubernetes Service):** AWS EKS was chosen to manage containerized applications efficiently. It provides a scalable and resilient environment for deploying microservices, enabling automated scaling and load balancing while simplifying Kubernetes management.
- **WAF (Web Application Firewall):** AWS WAF was deployed to enhance security by protecting the application from common web threats such as SQL injection and cross-site scripting (XSS). It helps ensure compliance with security standards and mitigates potential cyberattacks.
- **Cognito:** Amazon Cognito was integrated to manage authentication and authorization securely, since it will be used Azure ID of the company. It enables user sign-in, multi-factor authentication (MFA), and access control, ensuring a seamless and secure login experience for users across different regions.

Instead, Microsoft's DevOps tool was used to track and manage the project. A project was created on the tool and shared with the development team to track changes. DevOps contains the backlog and user stories that I wrote to list all the features required. It also provides a lot of collaborative solutions like comments, status tracking, tagging system, data attachments, linked elements, history, and other useful capabilities for developing.

In addition to serving as a repository for the backlog and user stories, Microsoft DevOps played a crucial role in workflow automation and task tracking. Each user story was broken down into tasks and assigned to the appropriate team members, ensuring clear accountability and efficient task management. The status tracking feature allowed for continuous visibility into the progress of each task, helping the team prioritize work effectively and quickly identify bottlenecks.

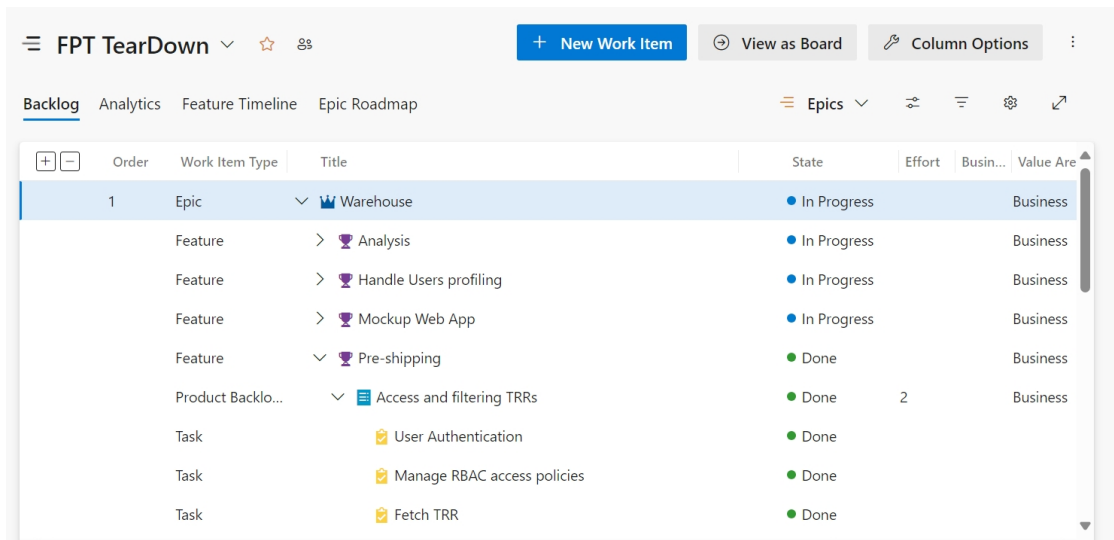


Figure 3.2: DevOps Teardown Project

Furthermore, integration with CI/CD pipelines facilitated automated builds and deployments, ensuring that new code changes were tested and deployed efficiently within the AWS environment. The ability to link pull requests, commits, and builds directly to user stories would help maintain traceability and provide a clear audit trail of development activities.

The commenting and tagging system improved collaboration by enabling clarifications within work items. Developers, testers, and product owners could exchange feedback, clarify requirements, and attach relevant documentation to streamline the development process. Additionally, the history tracking feature ensured that all modifications, decisions, and discussions were documented, making it easier to review past changes and track project evolution.

By leveraging these capabilities, Microsoft DevOps significantly enhanced team coordination, efficiency, and traceability, making it a central tool in the Agile development cycle. Its structured yet flexible approach to task management, issue tracking and automation helped ensure a smooth and transparent development process from start and will help maintainability.

For team communication, the company employed Microsoft Teams as the collaboration tool. The tool was needed for facilitating proper interaction among the members within different locations so that coordination and decision-making were made easier across the project.

Microsoft Teams was considerably used in holding many Agile ceremonies and meetings, including:

- Alignment meetings, during which stakeholders and team members discussed project goals, priorities, and the latest progress.
- Sprint Planning, where the team defined future work scope, assigned tasks, and set objectives for every sprint.
- Sprint Review, providing a structured meeting place to share work accomplished, receive feedback, and define subsequent requirements.
- Resolution sessions, aiming to resolve technical issues, discover blockers, and collaborate on resolution ideas.
- Comparison meetings, through which multiple architecture or implementation designs could be compared and good choices made.

Apart from scheduled meetings, Teams facilitated asynchronous communication, which allowed real-time conversations through chat channels, direct messages, and the sharing of files. The integration with other Microsoft 365 services, such as SharePoint, OneDrive, and Outlook, allowed the team to collaborate on documents efficiently, track changes, and maintain a centralized repository of project documents.

An Agile methodology was used, or at least it was expected. Specifically, the Scrum methodology was adopted for the project which was supported by my role as SM and the DevOps tool. I wrote that it was expected because the development team was not used to using it, and therefore, they needed my support with no small difficulties along the way.

To navigate these challenges, the development process has followed agile methodologies with continuous feedback loops, stakeholder involvement, and incremental improvements. Each requirement was carefully analyzed, broken down into manageable tasks, and iteratively refined to ensure a robust, scalable, and maintainable solution.

For the development of the application, the team chose Node.js for the backend and React.js for the frontend, leveraging their efficiency, scalability, and strong community support.

Node.js, a JavaScript runtime built on Chrome's V8 engine, was selected due to its asynchronous, event-driven architecture, making it well-suited for handling a high number of concurrent requests with minimal resource consumption. Its non-blocking I/O model ensures efficient real-time processing, which is crucial for applications requiring fast and scalable data handling. Additionally, its extensive ecosystem, including npm (Node Package Manager), provided access to numerous libraries that accelerated the development process.

On the frontend, React.js was chosen for its component-based architecture, which allows for a modular and reusable UI design. The use of the Virtual DOM enhances performance by minimizing direct manipulations of the actual DOM, leading to a smoother user experience. React's ability to efficiently manage state and render dynamic content was particularly beneficial in creating an interactive and responsive web interface for the Teardown Tool.

The combination of Node.js and React.js ensured a full-stack JavaScript environment, simplifying development by allowing both frontend and backend teams to work with the same language. This resulted in faster development cycles, easier debugging, and better maintainability. Furthermore, the choice of these technologies facilitated seamless integration with AWS services for hosting, authentication, and data management, ensuring a scalable and high-performance architecture.

Finally, Qualtrics was adopted as a survey system to collect structured feedback for analysis from users who interact with the Teardown tool. Qualtrics was selected due to its robust analytics capabilities, user-friendly interface, and integration options. The platform also supports conditional logic, advanced branching, and customizable question types.

3.3 Development and Implementation

3.3.1 Development Process

When I arrived, the project was at the beginning of the development phase. I was inserted into the corporate context and trying to become familiar with the colleagues who were dealing with the project.

Already during the first alignment calls with the supplier who deals with the development, a large part of the front-end was available. The various pages had been set up at a graphic level without yet connecting them to functioning logic.

I looked for feedback on the DevOps tool, and it was empty. I soon realized that the supplier was not used to working from an agile perspective. On the one hand, it was a plus because I would start from a blank canvas to structure the work on DevOps; on the other hand, it was necessary for the development team to also participate and contribute to its implementation.

I then created the basic skeleton of the project on DevOps, dividing the project into Epics, Features, and PBI (Product Backlog Item) based on the requirements of the product to be developed. In our case, there was only one Epic for now since the project, from an agile perspective, would follow Continuous Delivery (see Section 1.1), but it could have more.

I attempted to adhere to the organization's guidelines to frame the project within Microsoft DevOps in a manner that was consistent with the company's development standards. The structure is not, however, exclusively company-specific; it is common to many organizations because of its high level of effectiveness in coordinating Agile projects.

The framework is based on a hierarchical decomposition that facilitates the work organization efficiently and the monitoring of the progress in varying levels of detail:

- Epic – Represents a high-level business objective or a significant project milestone. Epics are broad initiatives that may take multiple sprints to complete and are further broken down into Features.
- Feature – A subset of an Epic that defines a specific functionality or component of the project. Features are designed to deliver tangible business value and are composed of multiple PBIs.
- PBI (Product Backlog Item) – Represents an individual requirement, such as a user story or a technical enhancement, that needs to be developed. PBIs are prioritized based on business needs and are the core of Agile development workflows.
- Task – A smaller unit of work that contributes to the completion of a PBI. Tasks are assigned to team members and define specific actions required for development, testing, or deployment.
- Bug – A defect or issue identified during development or testing that needs to be fixed. Bugs are logged, prioritized, and resolved to ensure software reliability and quality.

This structured approach supports easier monitoring of development progress, enhanced coordination of team members, and more openness in project implementation. By complying with this model, the team can efficiently plan, prioritize, and schedule workload, ensuring easy project delivery with high standards of quality.

Furthermore, this structure allows for maintaining the order and different sizes of granularity that still allows a low and high level vision even if the project is quite complex. At the beginning, it was very bare of content, with patience I managed to include all the user stories that were planned.

Every BPI tells a user story, more precisely in the BPI's *Description* box. In the BPI's *Acceptance Criteria* box, I wrote down a more technical and precise reference to what is needed to include in the application. This box is useful to developers to understand focal points of the BPI, writing them in a lower level understandable by programmers.

Starting from the Acceptance Criteria, developers should create child *Tasks* that contain single operations to perform, integration operations, or elements to include. These tasks represent the lowest level of functionality in the project and typically involve the simplest code to implement.

The development was then supported by several Microsoft Teams calls, where I provided guidance and training to the development team on how to properly use Microsoft DevOps. During these sessions, I explained how to structure tasks, ensuring they were clear, actionable, and aligned with the Acceptance Criteria. Additionally, I clarified the meaning of "Done" (see Section 1.1.3), defining the conditions that each task had to meet before being considered complete, including code quality standards, testing requirements, and team approval during Sprint Reviews.

Since there was not much familiarity with agile methodologies with the development team, and the alignment difficulties were evident, not all the ceremonies required by Scrum were followed. For example, *Daily Scrum* and *Sprint Retrospective* were substituted for a weekly light "*Teardown Alignment*" with main operational actors of the project (PO, SM, PMs).

The team must be aligned on the project developments from the very beginning, following the agile methodology. Therefore, I immediately organized weekly Sprint Planning and biweekly Sprint Reviews (once the Sprint has ended). Even if at the start there's not so much working to share, is it necessary to recap and revise the flow of usage of the tool under deployment. Sprint was defined as a 2-week time frame to achieve improvements and review in a short period.

I organized Sprint Planning with the PMs to maintain a hierarchy of decisions. During each meeting, PBIs were taken from the backlog and reread, adjusted, and clarified. Following these operations, they were estimated an Effort expressed in *MAN-DAY* (Man working days needed) based on the complexity of the implementation. The Effort could not exceed 4 days because otherwise it meant that the PBI was too large and needed to be broken into two or more PBIs.

Each person in the development team, which in our case was equal to one person for the back-end and one person for the front-end, was mapped by me on DevOps for the effort that they could dedicate. Since the two developers were exclusively committed to the project, and a Sprint corresponded to 14 days, the possible effort corresponded to 14 days each.

Each PBI was therefore assigned to the competent person when the *Capacity* was exhausted, thus defining the *Sprint Backlog*. It was then up to the developers to add the Tasks for each PBI and contact me in case of doubts.

It happened that at the end of the Sprint, some PBIs had not been respected in the deadlines, and that the DevOps tool was not used correctly because unrequested PBIs were being performed. This is a symptom that the developers did not access the tool because it was seen as something time-consuming: writing the tasks, consulting the Acceptance Criteria, etc., are part of the effort, and they do not understand that it is part of something bigger and need to follow a path.

Often, PBIs were re-inserted into the next Sprint, leading to iterations with very little development at a time, and Reviews spent more time on new ideas rather than confirming the work done.

One of the parts that make up a Scrum Iteration is testing. Unfortunately, this was not something possible for the first month and a half of development. The development team, not keeping in mind the agile principles, started with a waterfall-type development, developing the code locally on their own PCs. When they were asked to make the commits on the company repos because that should be the standard and, through a pipeline, deploy on the AWS environment, several permissions issues in the cloud environment came up.

Although in this case it is a shared fault, if the development team had followed all the necessary tests and precautions in the previous months, it would not have found itself in the development phase with impediments of this type. To solve these problems that popped up one after the other, I had to open several tickets, more or less elaborate, to support the infrastructure team. Since the FPT company is large and also has several applications internally, to open these tickets, it was necessary to be as precise as possible on the accounts, resources, and architectural design involved. Each opened ticket required a fair number of days to be resolved.

This situation has continued for quite some time; fortunately, it has not blocked local developments, but it has not favored the continued use of an agile methodology. Not being able to control, verify, and test the features has brought various problems later in time than they should have been.

When there was a chance to test in the development environment, some change requests emerged. These requests were handled via DevOps as additional User Stories and pushed into the next Sprint. In December 2024, two bigger changes came out:

1. APAC works slightly differently than EMEA, so requirements change as well;
2. For battery repair are required some additional notes are required.

These two changes of direction were not a problem thanks to the development method. They were inserted in detail in the backlog as User Stories, and the developments continued, taking into account future evolutions and preparing future additions. It was possible by maintaining a flexible structure of the database.

However, these major changes detected lead to the belief that the preliminary analysis phase was not fully explored and traced. Perhaps a more in-depth and debated analysis with the agile method, user stories, tracking requests and changes, would have kept more order and a general framework of the application that is intended to be developed. Not having been present at that stage, I cannot, however, comment on the work and take a strong position.

During the development phase, my role emerged several times with the role of keeping the reins of the work done so far and the developments still to be achieved. It often happened that the development team went a bit on its way, forgetting or not taking into account the requirements and/or user stories reported on DevOps.

An example of this is the case where it was necessary to add authentication to the web application and differentiate the permission levels and visibility of the users available in the application. The back-end developer was proceeding with the integration by creating an authentication function inside each lambda, without taking into account that the architectural design instead required the use of the AWS resource, Cognito. As previously reported (see Section 3.2.1), Cognito is built for this function, and its use allows maintainability, robustness, and resilience of the authentication method.

Thanks to my intervention, the second authentication strategy was implemented once all the configuration parameters requested by the relevant team had been obtained. Mine was not an imposition but a dialogue reporting the reasons why it was necessary to continue following a certain path.

The latest tests highlighted some problems with the functionality of various buttons, translations, or incorrect visibility permissions. Many small errors that could have emerged previously and saved time that had already been lost before with other hitches along the way. Certainly far fewer than those that would have emerged with a waterfall approach.

After 8 sprints, the application was ready to be launched in production in its first version. Therefore, it was required to replicate the resources in the production environment and all the permissions and connections that had been modified in the development environment to obtain a mirrored architecture.

At the time of writing, Turin's warehouse staff are piloting the Teardown Tool in a production environment. The several weeks of testing that have been ongoing will familiarize users with the web app, identify current bugs, and gather feedback on where improvements and new features are to be added in the future.

3.3.2 Development Technicalities

Teardown Tool development had to be carried out with a methodical approach to obtain scalability, security and efficiency. The technical implementation details are given in this section, with special focus on user profiling, workflow structuring, and warehouse management.

For seamless user interaction and access control, a user profiling system was adopted with varying roles and permissions according to organizational roles. They have been summarized in Table 3.1

User Type	TRR	Box	Item	Warehouse Position	Summary Analysis
Warehouse Admin User	All Regions	All Regions	All Regions	All Regions	Edit
FPT Warehouse Manager	All	Yes	Yes	Edit	Edit
FPT Quality / Warranty / Recovery Operator	Own	Yes	Yes	View	View
FPT Warehouse Logistic Operator	All	Yes	Yes	Edit	View
FPT Warehouse Teardown Operator	All	Yes	Yes	Edit	View
OEM/Dealer Operator	Own	No	No	No	View
Buffer Warehouse Operator	All	Yes	Yes	No	No

Table 3.1: List of available user types in the Teardown Tool

The messages reported are intended for the warehouse to which the user belongs, unless otherwise specified. A deeper level of permissions is given based on the role held.

User Authentication is entrusted to the existing Azure ID authentication infrastructure of FPT; while a second database-side profiling has been added to manually authorize the visibility that each user has, and therefore which role they should assume, in which region, and in which warehouse.

In addition, users who will have visibility of the access link to the tool on an

existing portal, the MyFPT Portal, which acts as a collector of company tools, will also need to be mapped. Based on the access performed on this portal, the links to the tools that they can use will be shown or not. Through a request managed with a ticket, a user can apply to be added to these systems.

FPT Teardown: Business Process

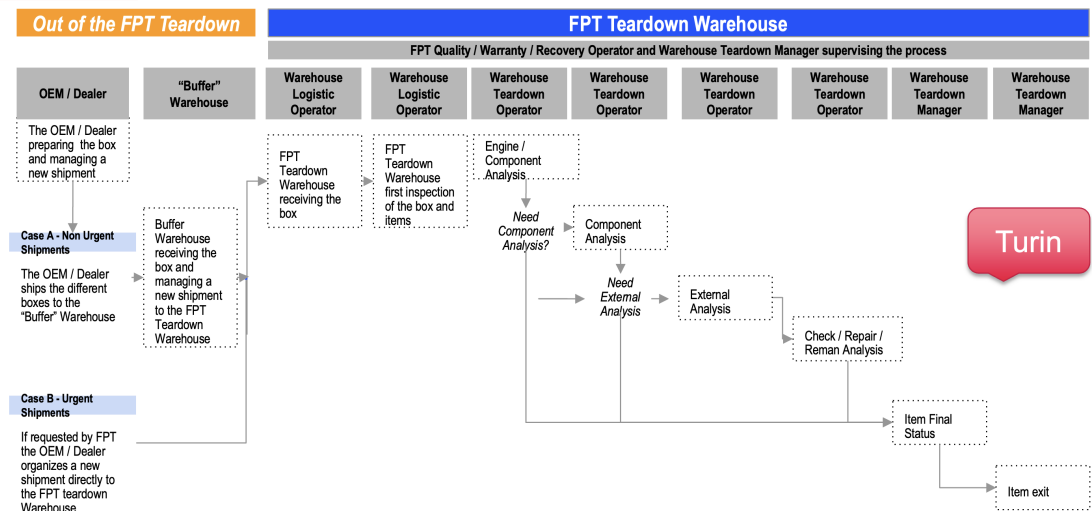


Figure 3.3: FPT Teardown Business Process for Turin Warehouse

The second phase was to define a Business Process for the objects that need to be handled. The following diagram (Figure 3.3) was designed starting from the operations that were already carried out by the Teardown team.

The same steps and operations were then reported in the tool that serves as a collector for the analysis and tracking of the statuses in which it is found. The possible statuses correspond to one of the phases described that can change automatically following operations conducted on the tool or manually.

The diagram also provides clarification on how the different user roles relate to each other and who is responsible for part of the flow. The process is designed to be streamlined and have multiple levels of responsibility. For example, the person conducting the engine/component analysis must be a qualified technician.

Teardown activity starts with the creation of a Teardown Return Request (TRR) added by the Dealer/OEM or by a Quality Operator. To open the request, you are immediately asked to enter the issue and a description of the issue. The request can then be saved and completed later. You can add one or more items to the request that can be uploaded individually or in bulk via a CSV generated by an

Excel spreadsheet downloadable from the same page.

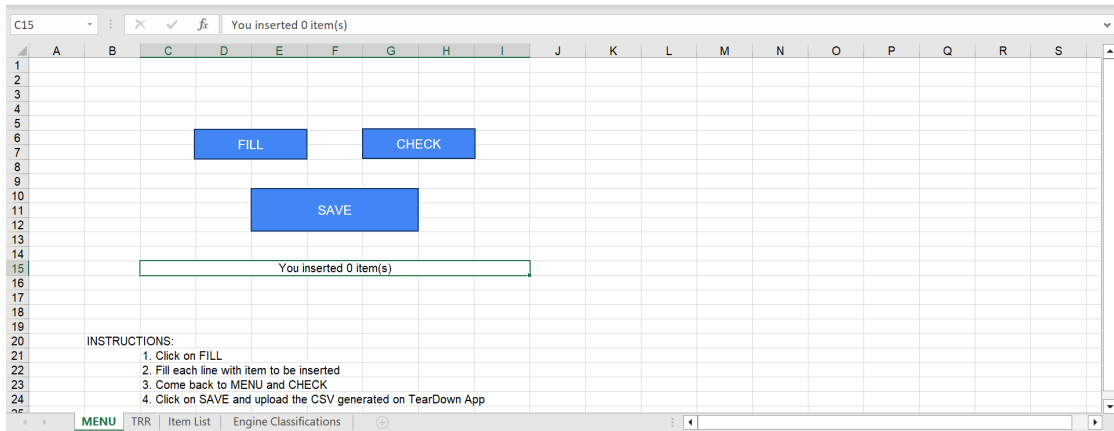


Figure 3.4: Menu and Instructions provided in the first sheet of the Excel

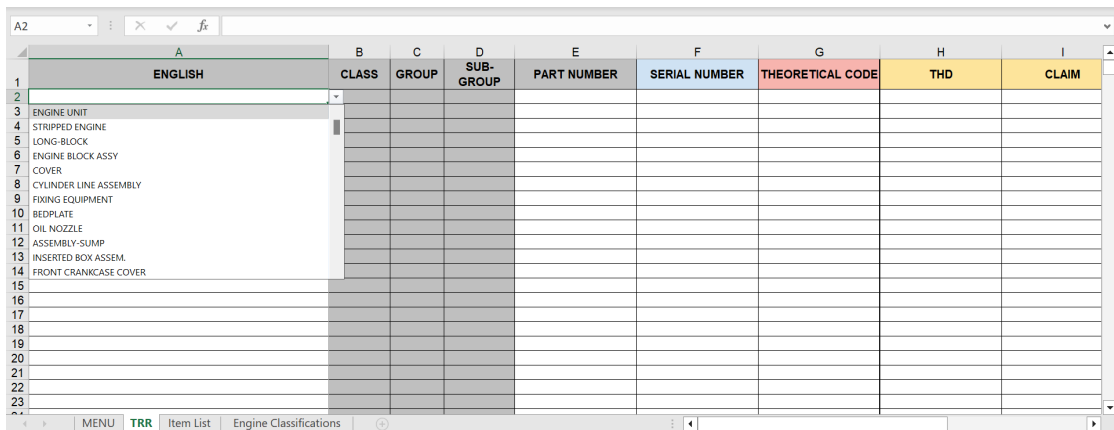
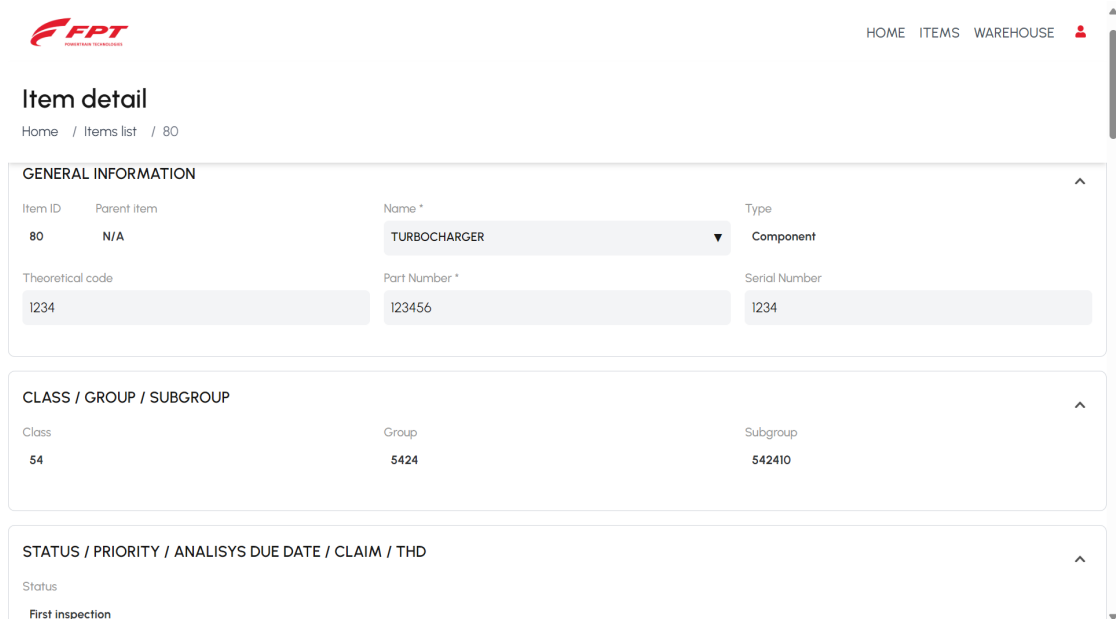


Figure 3.5: TRR definition with items selectable from a list in Excel for bulk import

The Excel file (Figures 3.4, 3.5) is available in different languages, automatically deducted from the browser's language. With some macros I designed, it is possible to fill, check, and save data for engine/components following some rules of compilation and obtaining useful data avoiding data errors. The spreadsheet guides the user in its compilation, and the inserted rules give timely feedback in case of errors.

The use of an Excel file for the mass import was considered since Dealers/OEMs were already asked for an Excel file with the list of objects that were inserted in a Teardown request. Familiarity with the use of Excel should favor the loading of the data requested for each object.



GENERAL INFORMATION

Item ID	Parent item	Name *	Type
80	N/A	TURBOCHARGER	Component

Theoretical code	Part Number *	Serial Number
1234	123456	1234

CLASS / GROUP / SUBGROUP

Class	Group	Subgroup
54	5424	542410

STATUS / PRIORITY / ANALISYS DUE DATE / CLAIM / THD

Status
First inspection

Figure 3.6: Item Details in Web Tool, general information and status

Once the request is filled, it can be sent, and data becomes permanent and no longer editable (Figure 3.6). At the receiving doors, the item is also tracked in its internal movement up to the Teardown Warehouse, where a Logistic Operator checks the content of the box, prints a QR Code for the box and items. If an item is missing, it could be marked as "not present" in this phase.

Logistic Operator can also attach photos and videos to the box and the items' arrival conditions. It's time to find them a place in the warehouse by selecting one of the available locations. In this way the warehouse will be kept in an orderly manner and will allow you to know in real time the position of each arrived object.

For the analysis, a survey was used that was already designed for use without the web app. The questions guide the Teardown Operator to identify the cause of the breakage or compromise based on the objective results found on the engine/component and produce analyzable data.

The tool set up for the web app instead changes shape by using the same set of questions, but is directly integrated into the developed tool to maintain a good User Experience and link the survey to an item in the database. It was asked to use Qualtrics as an integration which was a tool already present in the company and which promises easy integration.

The survey is structured with 180 questions that take into account, for example:

- general data on the Teardown Analysis, when it was conducted and by whom;

- identifiers such as serial number, part number, VIN, claim number, or THD;
- the engine family and its fuel;
- the characteristics of the vehicle on which it was mounted;
- the problem encountered;
- attach photos of the various analysis phases;

Since the survey is very long and should be meaningful, a lot of display logic is inserted to skip some questions given previous answers. It is also thought to auto-save answers up to the last page of the question answered. This would let the Operator start, pause, perform some analysis, resume the survey later, or let someone else resume it. Finally, a conclusive report in PDF is produced by these answers and attached to the item.

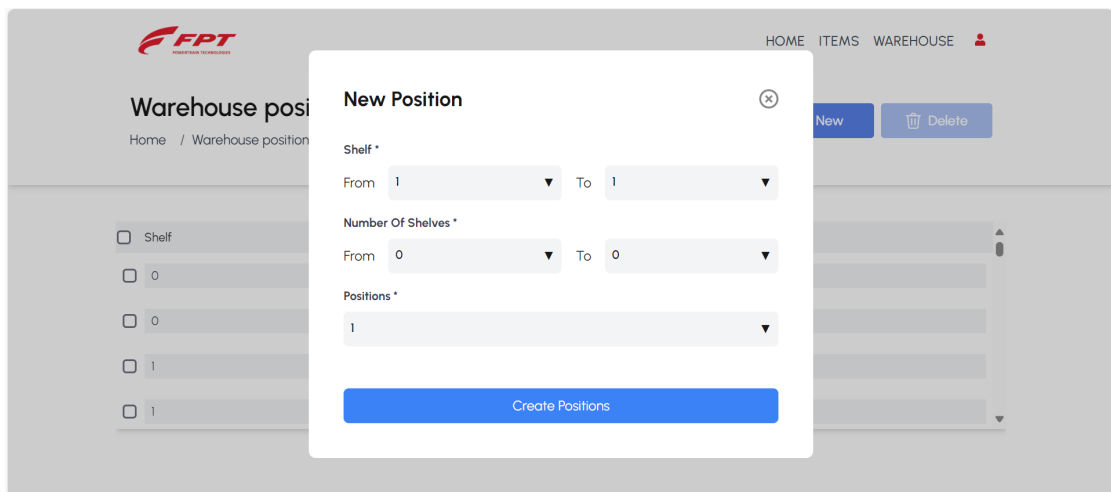


Figure 3.7: Web tool: Warehouse positions management

During the process, items are stored in the warehouse, and their position is also registered in the tool. To identify the position of an item, it has been implemented a triplet of numbers has been implemented: shelf, level, and position. Each warehouse has to be mapped by the FPT Warehouse Manager, creating all the different locations defined by a triplet. Thanks to its easy customization, the warehouse can change, evolve, split positions, or merge others to rethink the best fit for incoming items on demand.

The user interface (Figure 3.7) for creating locations is composed of a structure with "from ... to ..." shelves and levels and a precise number of positions. The tool

will create all storage spots with a nested loop. This implementation is useful to create, with a few clicks, a huge number of warehouse spots.

The tool keeps in memory and virtually occupies the position. When an item has to be shipped out, the position is automatically left free for something else. Under each item, details are prompted for all free positions in order of filling. It's also possible from a position, discover what an item is by scanning its QR Code.

Turin Warehouse has been mapped with 33 shelves, 2 positions for level 0 (floor), and 3 positions for level 1 and 2. The two shelves closest to *Technical Stands*, used for component analysis, have since been split into multiple positions because they are smaller than an engine, and each level can host 8 slots.



Figure 3.8: Turin Teardown Warehouse getting mapped

3.3.3 Challenges Faced

During the design and implementation process of the Teardown Tool, various challenges were encountered that needed vigilant problem-solving and coordination between various teams. These challenges were of a technical, organizational, and operational nature, thereby affecting the schedule as well as the project's efficiency.

VPN Problems

The company has a corporate VPN to protect communications and workstations while you are not physically in the company. The intranet network has greater privileges for some corporate tools. Some have incoming firewalls for protection, authentication, and authorization to access them.

It is a company policy to develop systems for internal use and therefore accessible only if under VPN. In the case of developing the Teardown Tool, the application must also be accessible to Dealers and OEMs and therefore exposed to the Internet. For the dev environment, it was decided to use a domain that can only be resolved under VPN.

In the dev environment, it is also possible to reach the database because it is created on an internal subnet to perform operations of creation, consultation and insertion of tables and tuples. VPN access is therefore essential to create the foundations of the project.

However, the development team did not test the access in advance, even though it had been provided to them months before my arrival, and found themselves blocked in the project because the connection did not work.

The first operation I suggested was to recheck the configuration provided by FPT for access as external consultants to the internal VPN network and, on our side, to verify that the access permissions that I had requested for certain IPs had been inserted in the firewall.

Having verified that everything was correct, but the problem persisted, I proceeded by contacting the company's ICT support, who, by checking the logs, was unable to trace any connection attempt. The suspicions then fell on the use of PCs with an operating system different from the company's and that perhaps company policies blocked it. This hypothesis was denied by a colleague of theirs who instead used the same VPN on another project.

After weeks of attempts and checks, I installed the VPN application on my PC to investigate whether there were any differences with their same operating system. From the logs generated on my PC, I saw that it checked for the presence of an antivirus. In a call with ICT and the developers, we finally managed to understand that the problem was dictated by the configuration of the security certificate that was generated by the device and which guarantees the reliability of the device.

An operating system update and a security check in the settings were enough to make the VPN work. The problem, which continued for weeks, however, led to a delay in developments on the AWS infrastructure, keeping everything locally. At that time, it was therefore not possible to test the first functions of the tool during the various Sprints.

Survey copy to Qualtrics

Teardown activities exist before the creation of the Teardown Tool. The survey analysis was already available and used to gain data and organize work. The problem is that it became necessary to move the survey from the current tool to Qualtrics.

The internal person assigned as a Qualtrics contact and who normally takes care of configuring surveys for the company, became aware of the length (180 questions, open-ended, multiple response options, etc.) and declared that he was not available due to time constraints to replicate all the questions by the end of the year.

To avoid having to wait and further prolong development times, I started looking for a smart solution that could streamline this process. I therefore studied the documentation of both survey platforms and developed a code in Python that would allow the survey to be exported in json format on one side and that the code I developed would take care of remapping the questions and loading them one by one in the various formats, with descriptions and answer options appropriately reconverted.

It was not a simple operation since the documentation provided by Qualtrics is fragmented, not updated, and often misleading. Through Trial and Failure, I was able to analyze the errors generated by the calls to the Qualtrics API and correct the calls to find a working solution.

Since the operation still took a lot of energy and time, the logic of visibility and jumping between questions was not copied through this automated procedure in Python. I asked to provide me with a PDF with the list of applied logics, and I manually copied, via Qualtrics' graphical interface, the various configurations.

This copying experience has produced:

- a code that can be useful for bringing other surveys to Qualtrics
- an awareness of the limited documentation available and therefore an acceleration of the Try and Failure times for the subsequent steps of integration with the tool
- a significant saving of time that waiting for the person in charge of manually copying this survey would have entailed

Security team and WAF configuration

Very late in the project, the security team requested to add the WAF configuration for the API Gateway and suggested adding a second one. This revealed an analysis problem that had been conducted perhaps in a not very thorough way and despite

the numerous meetings where an architecture had been agreed and confirmed, it now had to be modified again.

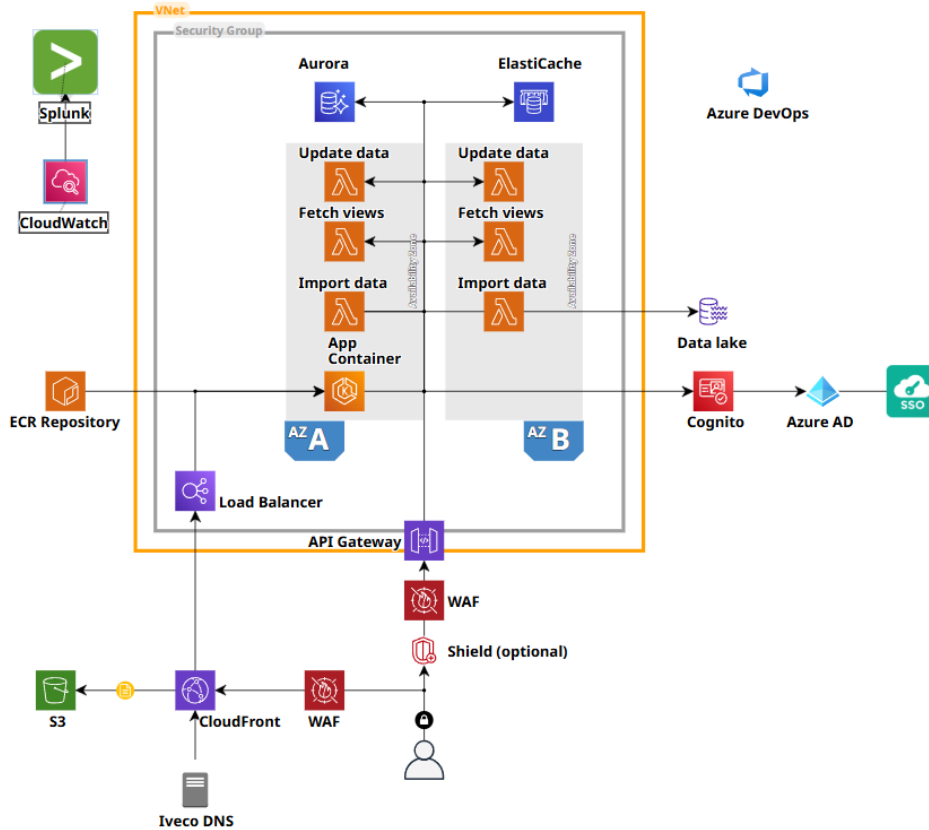


Figure 3.9: FPT Teardown Architecture (last version)

The symptom that can be found is that the methodology that had been adopted before and that brought about a great structural change at such an advanced stage of the project was not effective. Architecture has been modified according to Figure 3.9.

Furthermore, for the WAF configuration to block IP addresses or other potential attacks, an "allow" or "deny" rule was introduced. The rule applied by the security team stopped the functioning of the entire application even for those who were under VPN and who could legally access it. This is because it turned out that Cloudfront, due to its nature as a Content Delivery Network (CDN), is exposed on

a public IP address and to reach it, the VPN goes outside the range of internal IPs.

This rule, applied by the security team, blocked the use and development of the application for four days, during which they had to think of a solution and align with us in another call. When we got to the call, there was no ready solution, and they suggested changing the architecture again by removing Cloudfront at least for the development environment.

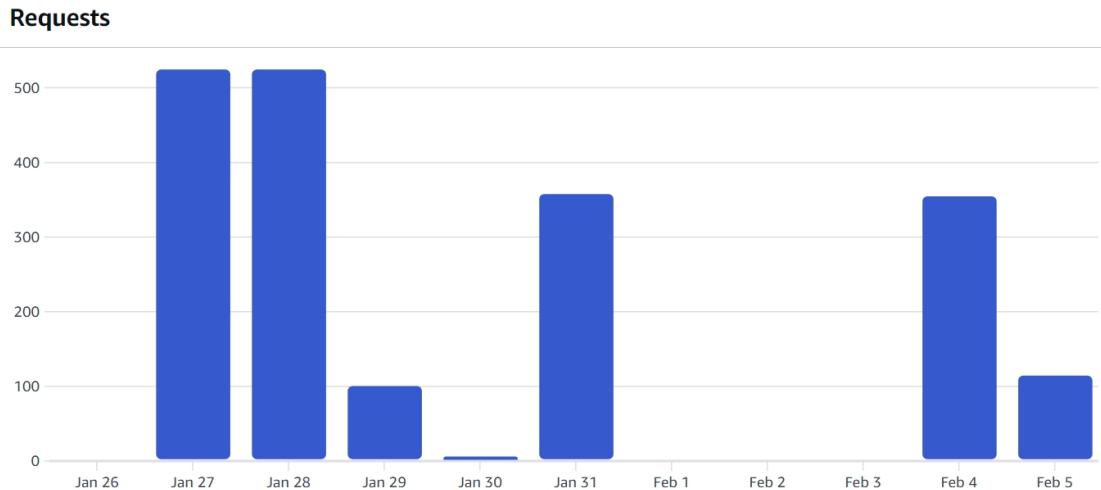


Figure 3.10: AWS Cloudfront WAF Allowed Requests

Following the call, I tried to write a rule for the WAF independently that took into account the impossibility of changing the architecture, because this would have caused costs and times for the implementation, slowing down the development. Furthermore, it also had to take into account making as few changes as possible on the application side.

What came out of my rule was a filter based on the Host of the origin request, which corresponds to the main domain to which the user connects. The domain, being resolved by internal DNS only after VPN connection, and with an SSL certificate issued by the company, still allowed a good level of security by filtering requests coming from other sources (Cloudfront remains contactable since it is public, but with an AWS IP or domain).

I contacted a person from the security team who was working on the case and explained my proposed solution. It was found interesting and was immediately tested live and solved the problem by only letting internal requests through, as expected.

Later the solution was checked with the security team, then upon its confirmation

of effectiveness, and directly added the rule on the Web Application Firewall (WAF). It permitted only internet request entries at the network level, thus enhancing the security of the system and preventing such issues from being created in the future.

The specific rule that was implemented as part of the solution is structured as follows:

```
{
  "Name": "test_header_match_teardownapp",
  "Priority": 1,
  "Action": {
    "Allow": {}
  },
  "VisibilityConfig": {
    "SampledRequestsEnabled": true,
    "CloudWatchMetricsEnabled": true,
    "MetricName": "test_header_match_teardownapp"
  },
  "Statement": {
    "ByteMatchStatement": {
      "FieldToMatch": {
        "SingleHeader": {
          "Name": "Host"
        }
      },
      "PositionalConstraint": "EXACTLY",
      "SearchString": "domain-name-host-teardown.com",
      "TextTransformations": [
        {
          "Type": "NONE",
          "Priority": 0
        }
      ]
    }
  }
}
```

Cyber Attack from abroad

More than a month later, I received information of an anomalous access from the Philippines with a credential used to deploy lambdas in an AWS environment. It was not anyone from the team, nor was there a VPN or proxy involved.

The user was blocked and the case of attempted access was investigated. From the logs for an in-depth analysis, it led to further attempts to access the accounts of the development team consultants from foreign locations such as Kenya, the USA, Brazil, etc. The logs also revealed that no command or diffusion was executed due to a lack of sufficient permissions.

As a precaution, the user was not recreated, and I suggested changing the password of the entire team to strengthen security. Before being able to recreate the credentials and also to start with production developments, the security team rightly wanted to understand how they had come into possession of the deployment credentials.

After about a week of investigation, the development team recalled that precisely following the previously mentioned problem of the WAF rule, since the downtime coincided with a Sprint Review, a copy of the backend and frontend had been pulled up on a system external to the company with an import of the secret that was exploited.

Following checks on the code committed to the repo, having agreed on the causes of the problem, the incident was closed, and new credentials were generated. The credentials were saved in the AWS Secret Manager and must be obtained from there each time to maintain greater security and avoid them being exposed even for just a few hours, as happened.

Most likely, during the hours for which this copy was online without a Firewall and WAF on an external system, some scraping bot arrived on the page and analyzed the JavaScript code available on the page to obtain useful information. From there the access key will have come out to connect as an IAM User to the AWS console of the dev environment.

This was another slowdown in development that happened during the co-occurrence of another event. Working safely should be another standard to follow that can be avoided when too many slowdown events occur, and a person is looking for the fastest way to get by.

Chapter 4

Conclusions

This chapter summarizes the key aspects of the project, including its objectives, achievements, challenges faced, and lessons learned. It provides insights into the implementation of Agile methodology within a software development team in an automotive company, offering suggestions for improvement. Additionally, it highlights the limitations of the current implementation and explores potential enhancements. The goal is to provide a comprehensive critique of the work conducted and its impact on the company.

4.1 Summary of the Work

This thesis embarked on a thorough examination of the development and the subsequent implementation of the Teardown Tool within the working environment of FPT.

The main objective of Teardown was the improvement of warehouse management procedures, a drastic improvement in the overall efficiency of teardown analysis, and eventually the delivery of a meticulously organized digital solution tailored specifically for the tracking and reporting of key information.

With the continually growing number of components and engines that need to be carefully analyzed, it was apparent that a more scalable and automated system was needed to effectively take the place of the initial manual procedures that were greatly dependent on spreadsheets and email exchange.

The process of development was characterized by the careful inclusion of cutting-edge cloud technologies, which was important in making it possible to scale, secure, and access a high number of regions.

Amazon Web Services (AWS) played a central role in providing the underlying infrastructure backbone necessary for this initiative. The software stack made use

of Node.js to scale backend services effectively and React.js to facilitate interactive frontend development.

Moreover, Agile methodologies were utilized purposefully, which enabled the team to maintain continuous feedback and change dynamically according to the ever-evolving business demands arising within the course of the project life cycle.

A key focus of this thesis was the introduction of Agile methodology within a development team that was not previously accustomed to using it. The transition required a structured approach to ensure a smooth adoption of Agile principles and practices.

As part of this project, my role was that of Scrum Master, where I played a key role in introducing the Agile methodology to the development team, which was not previously accustomed to its use. I initiated fundamental Agile ceremonies, such as Sprint Planning and Sprint Review, ensuring a structured yet flexible workflow. The development process was organized into two-week Sprints, each encompassing planning, development, testing, and final approval as "Done".

Aside from monitoring these iterations, I engaged myself in actively bridging the different teams' synergy and communication.

With the stakeholders' varied expertise from software developers, warehouse operators, to IT and security professionals, my role was crucial in bridging the technical and business requirements from one level to level of understanding.

This guaranteed that every team could function harmoniously without misunderstandings, coordinating their work towards a single aim. By acting as a connector between technical and non-technical disciplines, I facilitated smoother interactions, enabling effective application of Agile principles in the project.

My role frequently required me to act as a facilitator and problem solver, addressing challenges as they arose—whether by opening tickets, identifying viable solutions independently, or, when necessary, directly contributing to the codebase to resolve critical issues.

The collaborative effort with a broad set of teams, from customer care through to IT security, infrastructure, and warehouse management, was instrumental in the development and realization of the overall solution.

The active involvement and engagement of these stakeholders were important in ensuring that the tool not only addressed the key technical requirements but also aligned seamlessly with the various operational workflows and complied with the necessary compliance requirements.

By implementing this particular project, FPT has managed to establish a solid foundation that supports a more structured and systematic approach to teardown

analysis by using data. This significant advancement ultimately goes on to play a part in enhancing decision-making processes, hence the minimization of costs for warranties, in addition to fostering greater efficiency in areas such as quality control and warranty analysis.

Furthermore, the rollout of the Teardown Tool opens up new business opportunities since it enables the reuse of analyzed parts, thereby fostering a circular economy and contributing to the reduction of mechanical waste.

By systematically monitoring and assessing the condition of returned parts, FPT can weigh possibilities such as the refurbishing and sale of reusable parts, the recycling of materials or the optimization of recycling.

This not only supports sustainability efforts but also promises to deliver potential cost savings and revenue streams, reaffirming the company's dedication to environmental stewardship and resource efficiency.

4.2 Key Achievements and Results

The agile approach to application was strategic in the development and release of the Teardown Tool. By applying Agile principles, the project team was able to enable ongoing improvement, rapid response to shifting requirements, and effective collaboration across different departments. The following are the main achievements that demonstrate how Agile contributed to project success:

- **Iterative Development and Rapid Delivery.** The project utilized an iterative methodology with multiple sprints, allowing the team to provide working increments of the Teardown Tool in phases. This got essential functionalities completed, tested, and deployed faster, rather than waiting for a full release.
- **Successful Sprint Planning and Backlog Management.** With Microsoft's DevOps tool, the backlog was properly organized with Epics, Features, PBIs, and Tasks, Bugs were tracked too. Well-defined Acceptance Criteria enabled developers to comprehend requirements precisely, resulting in improved development effectiveness. Regular backlog grooming assisted in keeping the highest priority features ranked based on business need.
- **Flexibility in Adapting to Evolving Needs.** When new matters cropped up (e.g., security requirements, infrastructure changes), the Agile process facilitated quick scope and priority reordering.
- **Enhanced Quality Control and Bug Repair.** Testing was integrated into the development process, and bug-fixing and code review meetings were part of each sprint.

- Stakeholder Engagement and Continuous Feedback. Early end-user piloting (e.g., Turin warehouse pilots) debugged features before widespread deployment. End-user feedback from initial test cycles was incorporated into subsequent sprints and the tool therefore dealt with real working requirements.

Splitting tasks into smaller, comprehensible steps increased tracking and responsibility. Using Agile methods, the project team was able to create a very integrated and scalable solution that met business requirements, and it is open to future development. The iterative nature of Agile ensured that the Teardown Tool was improved based on real feedback, making it an incredibly valuable tool for warehouse management and teardown analysis at FPT.

I can say that even with several difficulties, the development team has entered the Agile world and acquired some new tools. It is not completely achieved as a goal because there are still some shortcomings that have caused slowdowns along the way and it has not been assimilated by all team members.

The same incidents and the work done before my arrival, however, are clear proof that a waterfall methodology cannot work in a complex business context in the case of software development. The use of an Agile method has allowed the rapid resolution of various problems that otherwise would have taken much longer.

4.3 Limitations of the Project

In this report, different project limitations have been recognized. With a closer look, the limitations can be linked to some important root causes that impacted some of the development and implementation aspects. It is imperative to determine the root causes so that I can understand potential areas of improvement for future versions of the Teardown Tool.

- Communication. Having chosen to adopt an Agile methodology, equally Agile communication should be used. Instead, communication with the development team has often been complex and fragmented due to the lack of a common and rapid tool for communicating.
- Concept of "Done". All teams must be involved and aligned during the development phases of the project, as it is their responsibility. It becomes a problem when, as far as they are concerned, a procedure, a development, or a decision is "Done", but after some time it is taken up again, reworked, and questioned again, causing the foundation of the house of cards to collapse.
- Guidelines. The lack of guidelines by the teams involved has caused an increase in entropy and the possibility of moving at will between the requirements. This

leads to "Try and Failure" that lengthens the time or a procedure accepted by a member of a team is not accepted by another member of the same team.

- Skills. The project proceeds quickly if the project team is well aligned, competent, and on the ball. It is not easy for this to be the case because, in a large company, you deal with different projects, and maintaining the same level of attention for everyone becomes difficult.

While these limitations posed challenges, they also provided valuable insights for future improvements.

4.4 Future Developments

For the Teardown Tool project, there are many possible points of evolution and expansion. There are at least three major areas of focus that should be developed and explored further:

- the alignment of the current tool to suit and operate optimally in another, more diversified region, including that of the Asia-Pacific (APAC) zone.
- the addition of an in-depth and elaborate analytical flow, especially made and customized for batteries
- a specialized section of the web application that is particularly structured to manage and coordinate the different incoming and outgoing shipment flows in a way that is much more effective and efficient

Further implementations are related to the connection with other company systems to verify the correctness of data entered and retrieve some useful records, such as the warranty expiration date, to control everything from a single place.

The developments that are ongoing will continue to comply with the Scrum approach, which has been a successful model for our team's work process. There is a good chance of adding more elements to the way to work as the team becomes more and more comfortable with this specific way of working.

For example, it might look into adding the Daily Scrum to the process, which enables more regular check-ins and discussions. In addition, the Board could be updated by the developers themselves directly to indicate Tasks that have been completed and are ready for verification and review.

4.5 Final Thoughts

Teardown Tool development has been challenging yet fulfilling. Not only did the project enhance the efficiency of warehouses, but it also demonstrated the power of

the Agile framework in the software development of the team's project.

Reflecting on this experience, I believe that my role as Scrum Master—typically associated with management engineering—was significantly enhanced by my background in computer engineering. While the Scrum Master is primarily responsible for facilitating Agile processes, ensuring team alignment, and removing obstacles, the ability to deeply understand the technical aspects of the project provided an invaluable advantage.

It allowed me to collaborate between development, infrastructure and security teams and speak in their technical language. It allowed me to effectively negotiate between different stakeholders. Instead of just relaying requirements or barriers, I was able to help solve problems, suggest effective technical solutions, and examine the consequences of decisions not just from a process standpoint, but from a software architecture and implementation standpoint as well.

This dual perspective—both methodological and technical—proved particularly useful in facilitating discussions, streamlining workflows, and promoting collaboration between teams with different specializations. It also enabled me to anticipate potential bottlenecks and proactively address them before they became critical issues.

Finally, I think that technical expertise combined with good organizational and leadership skills can go a long way in the success and performance of Agile projects. I also better understand the value of Agile methodologies not only as a methodology but as a philosophy of continuous improvement.

Bibliography

- [1] Kent Beck, Mike Beedle, Arie van Bennekum, et al. *Manifesto for Agile Software Development*. 2001. URL: <https://agilemanifesto.org> (cit. on pp. 1, 3).
- [2] J.A. Highsmith. *Agile Software Development Ecosystems*. Agile software development series. Addison-Wesley, 2002. ISBN: 9780201760439. URL: <https://books.google.it/books?id=uE4FGF0Hs2EC> (cit. on p. 2).
- [3] C. Jones and O. Bonsignour. *The Economics of Software Quality, Portable Documents*. Pearson Education, 2011. ISBN: 9780132564731. URL: <https://books.google.it/books?id=oEPjYVfUR1wC> (cit. on p. 2).
- [4] Kent Beck. *Extreme Programming Explained: Embrace Change*. Boston, MA: Addison-Wesley Professional, 2000. ISBN: 978-0201616415. URL: <https://books.google.it/books?id=G8EL4H4vf7UC> (cit. on p. 4).
- [5] Jeffrey K. Liker. *The Toyota Way: 14 Management Principles from the World's Greatest Manufacturer*. McGraw-Hill, 2004. ISBN: 9780071392310 (cit. on p. 5).
- [6] M. Poppendieck and T. Poppendieck. *Lean Software Development: An Agile Toolkit*. Agile Software Development Series. Pearson Education, 2003. ISBN: 9780133812961. URL: <https://books.google.it/books?id=IJ1gAgAAQBAJ> (cit. on p. 6).
- [7] Allen C. Ward, Jeffrey K. Liker, John J. Cristiano, and Durward K. Sobek II. «The Second Toyota Paradox: How Delaying Decisions Can Make Better Cars Faster». In: *Sloan Management Review* (1995). URL: <https://sloanreview.mit.edu/article/the-second-toyota-paradox-how-delaying-decisions-can-make-better-cars-faster/> (cit. on p. 7).
- [8] J.P. Womack, D.T. Jones, D. Roos, and Massachusetts Institute of Technology. *Machine that Changed the World*. Free Press paperback. Scribner, 1990. ISBN: 9780892563500. URL: https://books.google.it/books?id=_n5qRfaNv9AC (cit. on p. 7).

- [9] J.W. Evans and J.Y. Evans. *Product Integrity and Reliability in Design*. Springer London, 2011. ISBN: 9781447102533. URL: <https://books.google.it/books?id=IerjBwAAQBAJ> (cit. on p. 7).
- [10] F.P. Brooks. *The Mythical Man-month: Essays on Software Engineering*. Essays on software engineering. Addison-Wesley, 1995. ISBN: 9780201835953. URL: <https://books.google.it/books?id=fUYPAQAAMAAJ> (cit. on p. 7).
- [11] D.J. Anderson. *Kanban: Successful Evolutionary Change for Your Technology Business*. Blue Hole Press, 2010. ISBN: 9780984521401. URL: <https://books.google.it/books?id=RJOVUkfUWZkC> (cit. on pp. 8, 9).
- [12] M. Skarin. *Real-world Kanban: Do Less, Accomplish More with Lean Thinking*. Pragmatic programmers. Pragmatic Bookshelf, 2015. ISBN: 9781680500776. URL: <https://books.google.it/books?id=tol2rgEACAAJ> (cit. on p. 10).
- [13] J. Sutherland and J.J. Sutherland. *Scrum: The Art of Doing Twice the Work in Half the Time*. Crown, 2014. ISBN: 9780385346467. URL: <https://books.google.it/books?id=93tIAwAAQBAJ> (cit. on p. 11).
- [14] K.S. Rubin. *Essential Scrum: A Practical Guide to the Most Popular Agile Process*. Addison-Wesley signature series. Addison-Wesley, 2012. ISBN: 9780137043293. URL: <https://books.google.it/books?id=HkXX65VCZU4C> (cit. on p. 11).
- [15] Ken Schwaber and Jeff Sutherland. *The Scrum Guide: The Definitive Guide to Scrum: The Rules of the Game*. 2020. URL: <https://scrumguides.org/docs/scrumguide/v2020/2020-Scrum-Guide-US.pdf> (cit. on p. 12).
- [16] FPT Industrial. *FPT Industrial F28 Engine Awarded Diesel of the Year 2020*. 2020. URL: <https://www.fptindustrial.com/en/Pages/F28-DIESEL-OF-THE-YEAR-2020> (cit. on p. 15).
- [17] FPT Industrial. *Sustainable Truck of the Year 2021*. 2021. URL: <https://www.fptindustrial.com/it/media/company/prizes-accolades> (cit. on p. 16).
- [18] FPT Industrial. *Best Performer dell'Economia Circolare 2021*. 2021. URL: <https://economiecircolare.confindustria.it/confindustria-premia-i-best-performer-delleconomia-circolare-2021/> (cit. on p. 16).
- [19] FPT Industrial. *Alternative Engine Award 2024*. 2024. URL: <https://www.notiziariovi.com/news/15316/i-motori-a-combustione-interna-protagonisti-del-futuro-sostenibile-premiato-lxcursor-13-di-fpt-industrial> (cit. on p. 16).
- [20] Christopher Lovelock and Jochen Wirtz. *Services Marketing: People, Technology, Strategy*. World Scientific Publishing, 2016. ISBN: 9789814678195 (cit. on p. 17).

- [21] Philip Kotler and Kevin Lane Keller. *Marketing Management*. 16th. Pearson, 2022. ISBN: 9780135887154 (cit. on p. 17).
- [22] M. Savitha and Nagaraj G. Cholli. «Agile Methodologies In The Automotive Industry: A Comprehensive Survey». In: *IOSR Journal of Mechanical and Civil Engineering (IOSR-JMCE)* 21.5 (2024). DOI: 10.9790/1684-2105023035. URL: <https://www.iosrjournals.org/iosr-jmce/papers/vol21-issue5/Ser-2/D2105023035.pdf> (cit. on p. 23).
- [23] Denny Jean Cross Sihombing. «Improving Warehouse Operations in Cargo Companies through Agile-based Warehouse Management System». In: *Jurnal Ekonomi* 13.01 (2024). DOI: 10.54209/ekonomi.v13i01. URL: <https://www.ejournal.seaninstitute.or.id/index.php/Ekonomi/article/view/4245/3448> (cit. on pp. 24, 25).
- [24] Ralph Hughes. *Agile Data Warehousing Project Management: Business Intelligence Systems Using Scrum*. Newnes, 2012. ISBN: 9780123964632. URL: <https://books.google.it/books?id=2DC53dWqKBEC> (cit. on p. 25).
- [25] Brian Katumba and Eric Knauss. «Agile development in automotive software development: Challenges and opportunities». In: *Product-Focused Software Process Improvement: 15th International Conference, PROFES 2014, Helsinki, Finland, December 10-12, 2014. Proceedings 15*. Springer, 2014 (cit. on p. 27).
- [26] Robert K. Yin. *Case Study Research: Design and Methods*. Vol. 5. Sage Publications, 2009 (cit. on p. 27).
- [27] Lucas Gren and Martin Shepperd. «Problem reports and team maturity in agile automotive software development». In: *Proceedings of the 15th International Conference on Cooperative and Human Aspects of Software Engineering*. 2022 (cit. on p. 28).
- [28] N. Moe, T. Dingsøy, and T. Dybå. «Understanding self-organizing teams in agile software development». In: *19th Australian Conference on Software Engineering (ASWEC'08)*. IEEE, 2008. DOI: 10.1109/ASWEC.2008.4483195 (cit. on p. 28).
- [29] D. Hodgson and L. Briand. «Controlling the uncontrollable: ‘Agile’ teams and illusions of autonomy in creative work». In: *Work, Employment and Society* 27.2 (2013). DOI: 10.1177/0950017012460315 (cit. on p. 28).