UMinho | 2016

Renato da Cunha Castro

Automotive HMI: Management of
product development using Agile framework

Universidade do Minho
Escola de Engenharia

Renato da Cunha Castro

Automotive HMI: Management of product development using Agile framework

October 2016

Universidade do Minho
Escola de Engenharia

Renato da Cunha Castro

Automotive HMI: Management of product development using Agile framework

Master Thesis
MSc on Industrial Electronics and Computer Engineering

Supervised by
Professor Jorge Miguel Nunes dos Santos Cabral

October 2016

“It’s not that I’m so smart, it’s just that I stay with problems longer”
Albert Einstein

ACKNOWLEDGMENTS

Words cannot express my gratefulness for the help and encouragement received along this journey. This achievement is also result of the inspiration and support of many important people, who I must remember and truly thank.

To my advisor, professor Jorge Cabral for the availability to monitor this work, and specially for the motivation on pursuing my personal interests as a future engineer.

To my mentor and friend, Marco Martins, for the opportunity and encouragement on embracing new challenges. Thank you for making me grow!

To my investigation teammates for their support, but mainly for the fellowship and the laughs which definitely made the work fun and enjoyable.

To my insuperable colleagues of electronics engineering. We battled together for years! Despite following different paths, our experiences will never be forgotten! A special recognition to my partner and good friend Pedro Silva, for his support and encouragement along these years.

Finally, to my family and friends, for the patient and support, even though they still have no idea what I have been doing in the past ages. Yes, studying is no longer an excuse for missing dinners!

Thank you all, for the learnings, for the laughs, for the challenges ... For shaping who I am!

All the best!

ABSTRACT

This work aims to investigate the compliance and potential benefits of applying Agile to automotive development. Technological innovation has had a strong impact on recent decades with a major focus on the automotive world. A growing amount of devices are connected to the car leading to a sharp increase of available functionalities, which are expected to grow in number and in complexity over the next few years. Therefore, in order to keep pace with technological growth, a constant renewal of human machine interface systems is required leading to a considerable decrease of the period of product life cycle in the automotive industry. Consequently, with the purpose of responding to competitiveness and matching the user needs, it is mandatory for the automotive world to adopt new development methods in order not only to manage the growing complexity but also to reduce the time to market, since this readiness is crucial to maximize the return of investment. The proposed solution aims to meet this necessity through the use of Agile methodologies, focused on iterative development and oriented to customer needs. Thus, an action research was conducted aiming to evaluate the efficiency and compliance between the framework and the automotive industry. After an initial study on Agile methods, a process was designed for an automotive development project, in partnership with a reputable company in the automotive industry. Data gathered along this case study shown the major benefits and drawbacks of employing the Agile into a development project. Finally, the implemented approach was matched with recognizable models as CMMI and ASPICE, revealing the Agile compliance for automotive industry. (Davydov, 2012; "Manifesto for Agile Software Development," 2001; Oracle & Paper, 2013)

KEYWORDS: Automotive Development; Agile; Compliance;

RESUMO

O presente trabalho pretende investigar a compatibilidade e potenciais benefícios de aplicar metodologias Agile ao desenvolvimento automóvel. Ao longo das últimas décadas, a inovação tecnológica tem causado um forte impacto no ramo automóvel. O aumento de dispositivos conectados ao carro é refletido no elevado crescimento de funcionalidades, que tendem a crescer em número e complexidade ao longo dos próximos anos. Para acompanhar o crescimento tecnológico, o ramo automóvel é obrigado a uma renovação constante dos interfaces homem máquina, conduzindo a uma redução considerável no ciclo dos seus produtos. A fim de manter a competitividade e responder às necessidades dos utilizadores, o ramo automóvel carece de novos métodos de desenvolvimento. Além de lidar com a crescente complexidade, a nova abordagem visaria também a redução do tempo de mercado, que é crucial para maximizar o retorno de investimento. A solução proposta aborda estes desafios através da aplicação de metodologias Agile, centradas em desenvolvimento iterativo e orientado às necessidades do utilizador. Uma investigação ativa foi conduzida com o objetivo de avaliar a eficácia e compatibilidade da *framework* com a indústria automóvel. Após um estudo inicial sobre os métodos Agile, um processo foi concebido para um projeto de desenvolvimento automóvel, em parceria com uma respeitável companhia nesta indústria. Evidencias recolhidas ao longo deste caso de estudo mostraram os potenciais benefícios de aplicar Agile num projeto de desenvolvimento. Por fim, o processo foi comparado com modelos como CMMI e ASPICE, expondo a compatibilidade entre Agile e a indústria automóvel. (Davydov, 2012; "Manifesto for Agile Software Development," 2001; Oracle & Paper, 2013)

PALAVRAS CHAVE: Desenvolvimento Automóvel; Agile; Compatibilidade;

CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACRONYMS DICTIONARY

HMI	Human Machine Interfaces
ECU	Electronic Control Unit
TTM	Time To Market
ROI	Return Of Investment
AR	Action Research
IT	Information Technology
DSDM	Dynamic Systems Development Method
FDD	Feature-Driven Development
CI	Continuous Integration
VCS	Version Control Software
TDD	Test Driven Development
XP	eXtreme Programming
ASD	Adaptive Software Development
JIT	Just-In-Time
FDD	Feature Driven Development
DoD	Definition of Done
CMMI	Capability Maturity Model Integration
SPICE	Software Process Improvement and Capability dEtermination
ASPICE	Automotive SPICE
LCP	Life Cycle Processes
PA	Process Attribute
DSM	Driving Simulator Mockup
COTS	Commercial Off-The-Shelf

1. INTRODUCTION

This preclusive chapter intents to firstly describe the scope of the project and its relevance for the automotive industry. Furthermore, it includes a brief explanation of the research methodology and an overview of the document layout.

1.1 BACKGROUND

Over the past decades, the global impact caused by technological revolution is undeniable. The effects are evident in an increasingly consumerist society, always looking for the ultimate novelty. Although the tendency is verified in several domains, this constant effort to innovate always had a special focus on the automotive world.

Efforts have been made to push mechanical performance to the limits, conceive futuristic and pleasing designs, find novel alternative materials, and even attempt new energy sources. However, the greatest improvements are definitely inside the vehicle. Nowadays the common driver can take advantage of a large number of infotainment features in order to make the driving act more safe and comfortable. These functionalities became so popular that today they are widely available, proving that consumers are willing to pay for technology that enhances the driving experience [1].

The tasks performed by drivers are mostly travel-related, so the Human Machine Interface (HMI) is centered over traditional functionalities regarding vehicle diagnosis and telematics, navigation, traffic and weather information, etc. However, as consumer electronics like smartphones and wearables constantly evolve and gets connected, users expect automotive technology to do the same [2]. Therefore, automotive HMI is including concepts and technologies well known from personal devices: traditional buttons and visual warnings are being replaced by touch, speech commands, gesture recognition, and biometric sensing.

Thus, it is not a surprise that, in order to meet market requirements and achieve the desired interoperability, the electronic content in regular cars has drastically raised. A decade ago, it represented about 20 to 30 percent of production costs, but that number is expected to double in the following years [1]. While the average car nowadays has nearly 25 microprocessor-based electronic control units (ECU's), some premium models already surpassed 100 independent modules [3]. Such a sharp growth is explained by the fact that more than 80% of automotive innovations are now related to software [4]. Thereby, including a new functionality directly results in a considerable increment of code; so the amount of software on cars has reached impressive numbers. Although complexity should not be measured through the number of code lines, they might provide a general impression of the system, as depicted on chart below.

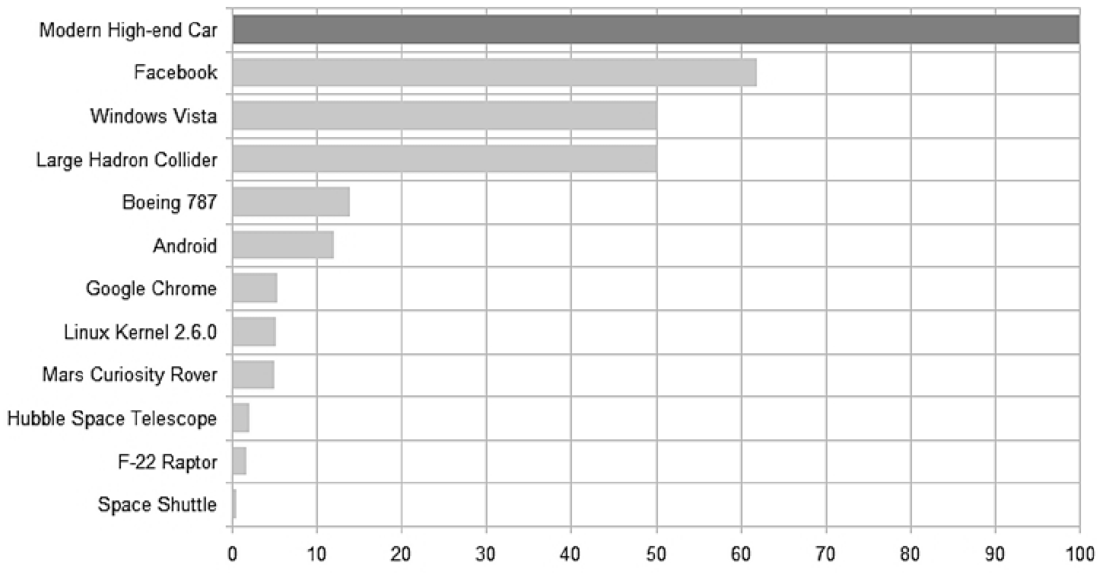


Figure 1 - Software Size: Millions of Lines [5]

Despite the global concerning about software and its importance within the automotive industry, few people know that their own cars contain the double of software volume of Windows Vista Operating System, or 8 times more than a Boeing 787. Figure 1 shows that a modern car runs around 100 million code lines, and this number is expected to increase to 200-300 millions in the near future [5].

Besides the volume of the software, its structure is also evolving. Since the beginning of automotive industry, the concern always was to define a new car functionality as independent as possible, so their development and production could be modular. Nevertheless, the sharp increase of software-based functions is clearly breaking that independence [6]. As a result, the same car that once was pure modular assembly now has to be understood as a complex system where all the software functions act together.

This central and increasingly complex role of software in cars brought several challenges to automotive development, especially when it comes to innovation. At the same time as software-realized functionalities increase in number, quality demands for reliability, safety and performance remain high. Moreover, literature reveals that each time a new car model is conceived, more than 90% of the software must be rewritten [7].

Consequently, to introduce a new product into the market is not only expensive (can cost up to $6 billion [8]) but it is also a long and extensive process. When compared to other industries and its products such as smartphones or computers, automotive has a time to market (TTM) almost four times higher, as depicted on Figure 2.

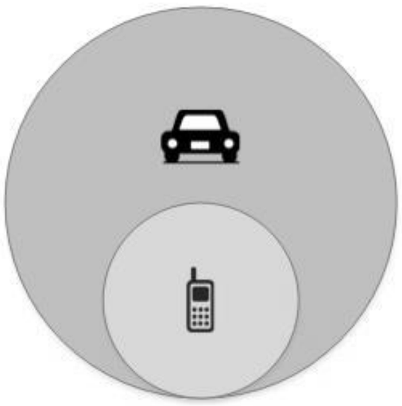


Figure 2 - TTM Comparison: Car vs. Smartphone

Considering the constant pressure to innovate and keep pace with other technological industries, automotive finds here a resilient obstacle. Buying preferences concerning automotive are expected to change like never before [1], so automotive industry must be prepared to react to market demands. Thus, release products faster is absolutely crucial in order to capture their full lifecycle and realize the desired return of investment (ROI).

Therefore, competitive advantage within the automotive industry critically depends on finding a solution to manage innovation and time to market. Since software is becoming a core activity in that extensive process, the key might be on software development methods.

1.2 Motivation and Objectives

From the software engineering perspective, the automotive industry is dangerous as well as a fascinating domain. Besides being an area characterized by a high investment risk, automotive is facing increasing complexity, and the pressure to not only innovate, but innovate faster. Flee from these challenges would mean be unresponsive to business conditions, so automotive urgently needs a solution to surpass them.

Analysing other domains where software performs an important role, an answer may be found through the use of Agile processes. As a flexible and iterative framework, Agile focuses on continuous delivery, customer collaboration, creative teamwork, and the ability of responding to change. These values intend to accelerate the development, always maintaining the high quality of the product. Since its official presentation, Agile has been successfully implemented by several software companies [9][10][11], attesting its potential benefits. Although automotive tends to progressively embrace practices from other industries, Agile still remains distant from automotive development. In fact, there is little research about how the framework can be applied to this particular domain, and consequent lack of results on the impact of Agile implementation on an automotive development project. This thesis aims to fill that gap and explore a solution for automotive urgent needs through the use of Agile methodologies.

In such a broad objective, the initial step is to conduct a detailed study on Agile principles, methods, and core practices. Since the process targets an automotive development environment, the research shall also address automotive norms and standards to comply with. From this study and critical analysis, is expected to design an Agile framework which should ensure the operation of all stages of a development project. The conceived approach shall then be applied to an HMI automotive development project conducted in partnership with Bosch Car Multimedia, entitled “Innovcar: The Cockpit of the Future”. The main purpose of the project (further described in 3.1), is to develop innovative HMI solutions for futuristic cars. Therefore, in addition to a single opportunity to increase the knowledge about the automotive development, this project is the perfect experiment for the hypothesis formulated by this study. Along with data gathered along the implementation of the process, conducted appraisals shall reveal benefits and drawbacks of employing Agile and its compliance with automotive development.

Summarizing, the purpose of the present work may be decomposed on the following goals:

- Research on Agile framework focusing its methods, principles, and core practices;
- Design and implement an Agile process into an HMI automotive development project;
- Match and evaluate compliance of implemented process with automotive development standards;

Since automotive development is a highly extensive domain, the scope of the study will be delimited in accordance with the requirements of an experimental project. Nevertheless, the final results shall provide general guidelines for the application of Agile methodologies on other development environments, as concrete information on its compliance for automotive development.

1.3 Research Methodology

In order to plan and establish a method for research, it is essential to define and clarify its key questions. Although the main focus is on the automotive challenges previously mentioned, it would be unrealistic to cover all domains of such a broad and complex subject.

Therefore, considering both time and project constraints, the scope of this research was narrowed to the following questions:

- How can Agile be applied to a specific automotive HMI development project?
- Is Agile compliant with automotive development?

As a solid research needs to be accurately grounded, the first phase of the method consists in doing a theoretical study about Agile practices and its suitability for automotive development. After defining an Agile approach, it will be tested on a real automotive HMI development project in which the author has a decisive role as product manager. Throughout the development process, data will be gathered in order to clarify advantages and disadvantages of using Agile methodologies in this environment. Moreover, the implemented process shall be appraised and matched with models for development, attesting Agile compliance with automotive standards.

In order to address both scientific research and active part on the project, an Action Research (AR) methodology will be employed. Widely used in information systems research [12], AR is an approach where the investigator forsakes the traditional role as observer and takes part in the real situation [13].

Besides helping an organization solve its problems and improve productivity and the quality of their products, AR involves gathering, analysing and drawing conclusions from research data [14].

Therefore, AR is a method that both solves an immediate practical problem while developing scientific knowledge. In this particular case, it is expected to evaluate Agile as possible solution to automotive challenges through an active participation on an automotive HMI development project.

AR methodology is driven over five cyclical phases:

- Diagnosis: Identify the primary problem;
- Action Planning: Determine actions that should relieve or improve the real problem;
- Taking Action: Implementation of the plan through collaboration between researcher and other practitioners;
- Evaluation: Determination if the previous action produced the theoretically expected results and whether these relieved the problem.
- Specifying Learning: Final conclusions and formalization of the knowledge obtained along the AR.

Since it is an empirical procedure, the goal is to continuously improve the approach and some practices might be changed along the process. Figure 3 presents the stages of an Action Research method.

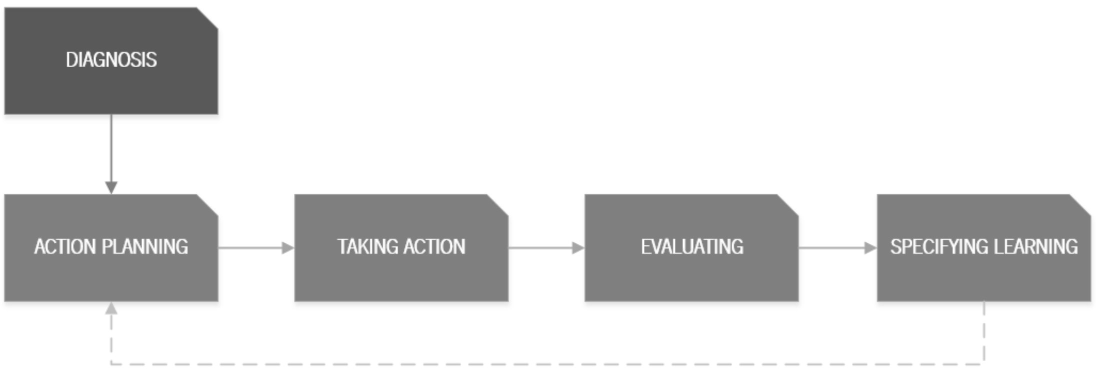


Figure 3 - Action Research - Five stage process

Since this study serves both scientific and business sides, it is necessary to consider the practical goals of the client and the research goals of the scientific community. Thus, the major objective is to design a successful solution for this particular development project always targeting the big world of automotive.

1.4 Document Layout

Following the structure delineated by previous methodology, this document is divided in 5 chapters:

- 1 “Introduction”: An opening chapter addressing the major concerts of automotive development and its main challenges, followed by the research questions which drove this study. Moreover, the research questions are stated as the implemented methodology to seek their answer.
- 2 “Literature Review”: As the base of the overall work, it presents all theoretical concepts needed to design the development approach. Initially, it is described the evolution of automotive development processes which led to the present challenges. Then follows a close focus on Agile development methods and practices. Finally, the major automotive development standards are addressed, constituting the basis for a further comparison with Agile.
- 3 “Methodology”: After the theoretical study, it is necessary to perform the *’action planning’* and design the Agile methodology to apply on the HMI development project. Firstly, this chapter provides a detailed explanation about the project. Then, it presents the Agile process to be implemented on the project, and relevant practices from the product management perspective. Finally, the chapter describes the method to appraise the process according to automotive norms.
- 4 “Results and Discussion”: Correspondent to the evaluation phase, this chapter exhibits the results of the employed methodology. A practical example provides a clear insight on the implemented process. Moreover, appraisal results evince the process compliance with automotive development models.
- 5 “Final Conclusions”: Considering the presented results, this final chapter contains the final conclusions focusing on the impact of Agile on the project and how that may represent significant findings to the automotive industry.

2. LITERATURE REVIEW

This chapter addresses relevant concepts to design the development approach. Primarily it describes the evolution of automotive development processes. The aim for a solution introduces Agile, focusing its methods and practices. Finally, the main automotive development standards and models are addressed, constituting a basis for a further appraisal.

2.1 EVOLUTION OF AUTOMOTIVE DEVELOPMENT PROCESSES

Nowadays, hundred millions of code lines, associated to a high amount of ECU's [15], are loaded into every car. However, it has been a long time since the first ECU was introduced to automotive world. General Motors was the pioneer when, back in 1981, introduced the first successful unit to the market [16]. Nevertheless, software was not so complex neither demanding as now; so the development process was not the main concern. Moreover, the initial guidelines for development could be provided by other industries which already had solid experience on software.

Therefore, the early approaches of automotive software development were based on traditional methodologies. In this category, the most recognized models are Waterfall and V-Model, posteriorly described in detail. However, due to some constraints and limitations, software development is evolving to more flexible and iterative approaches, already adopted in several areas by companies all over the world.

Despite being characterized by slower changes, automotive world seems to be closer to that mindset. The evolution of its software development processes described along the following sections shall make it clear.

2.1.1 WATERFALL MODEL

Based on a sequential development process, Waterfall provided the primordial solution for larger projects. Consequently, since the first formal description in 1970 [17], this model has been widely used in several domains.

In a simple view, development is seen as flowing steadily downward through several phases, as shown through the Figure 4.

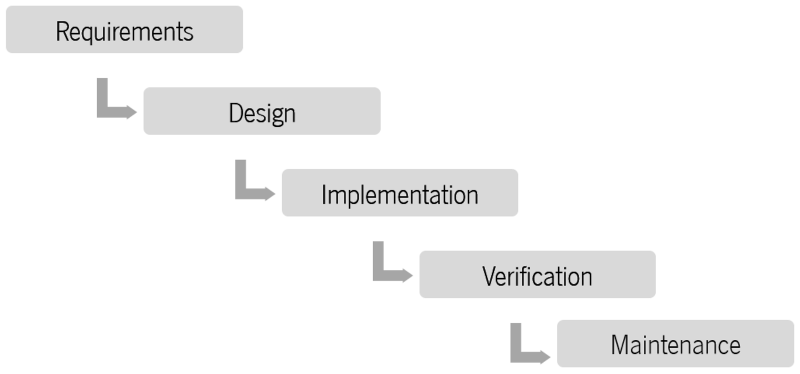


Figure 4 - Waterfall Model phases

The sequential phases in Waterfall model include:

- Requirements: Gathering and analysis of all possible requirements of system to be developed in order to produce a complete specification document.
- Design: Study of requirements and overall system architecture definition, including hardware and software specification.
- Implementation: Development of system based on inputs from previous phases. Typically, implementation is made through small programs to simplify functionality testing.
- Verification: Tests on each unit and finally, the integration of the whole system.
- Maintenance: Support and deliver of improved versions to customer environment.

Besides the clear definition of each phase, the overall process is simple and organized, easing its understanding and implementation.

The highest concern of Waterfall methodology is definitely to achieve a solid requirement gathering and planning stage. That initial effort intends to reduce time spent in later stages, especially during implementation. Another important focus is to produce extensive and high detailed documentation about every stage of the process. Meticulous written guidelines might be advantageous when new developers join the project. Characterized by strict and rigid values, Waterfall methods allow a certain predictability on time and cost estimations, establishing a clearer view of the whole project.

After long decades of test and implementation, Waterfall Model definitely has its place on software engineering history and proved that can be a useful solution for several kind of projects.

2.1.2 V-MODEL

Since its first presentation in 1981 [18], V-Model became increasingly popular until being considered the most used process within automotive industry nowadays. Also known as Verification and Validation model, it is based on association between test and development phases which are disposed in a V shape.

As an extension to Waterfall, V-Model is equally a sequential development life cycle process, so the stages are essentially the same. However, as the V shape reveals, the procedure is divided between two main cycles: development on left and validation on right, united by coding phase on the middle. Then, for every development phase a corresponding testing phase should be planned in parallel.

The working principle of V-Model is clarified by the Figure 5, presented below.

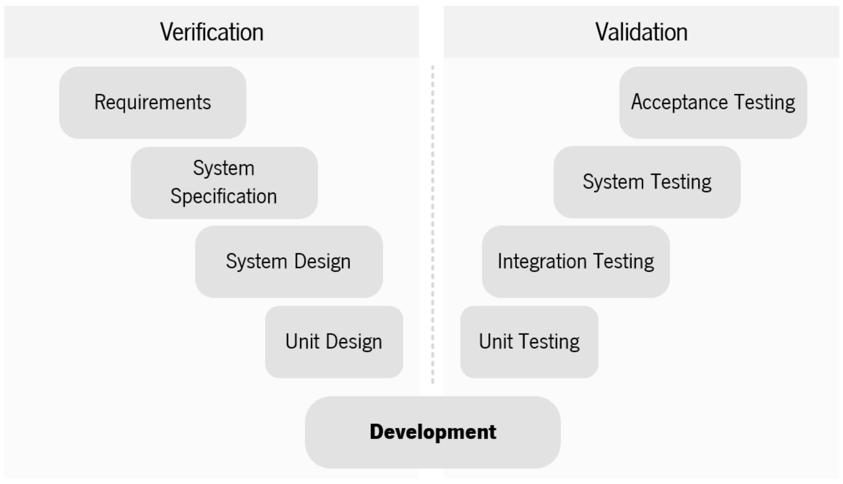


Figure 5 - V-Model process

Verification sets the beginning steps of the process through the following phases:

- Requirements: Evaluation of user needs in order to establish what features should be included in the final system. All requirements imposed by the customer such as interface, performance and security are compiled into a detailed document which has an important role to system design. As mentioned before, verification phases should also plan the correspondent validation stages. Therefore, at this point acceptance tests must also be designed based on specified user requirements.
- System Specifications: Study and analysis of user requirements for the purpose of define techniques to solve the proposed problem. Thus, a software specification involving system organization, data structures and interface menus should be the content of a new document which serves as a blueprint for development stage. Finally, metrics for system testing must be prepared.
- System Design: Also referred as High-Level design, this stage is responsible for the design of the global system's software architecture. Consequently, at this point overall concepts like architectural diagrams, list of modules, interface relationships, dependencies, and databases should be detailed. Lastly, integration tests are also designed at this phase.
- Unit Design: Described as a Low-Level design, this phase implicates a division and explanation of the global system into smaller modules, facilitating implementation of individual units. Following any programming convention such as pseudocode, every module should contain details about database tables, dependency issues, error messages, and a complete list of inputs and outputs. Ultimately, each development unit should be tested separately.

A complete and detailed verification provides a clear roadmap to the implementation phase, when the system is finally implemented. Development is followed by a long process of validation, correspondent to the second cycle of V-Model. Every validation phase has been already planned in parallel with verification, facilitating the course of the next steps:

- Unit Testing: Plans executed to obliterate bugs at code or module level. Basically, a unit test verifies if the smallest entities function properly when isolated from the whole system.
- Integration Testing: With plans conceived during system design, integration tests verify if units are able to communicate and coexist among themselves.

- System Testing: Differently from previous verification tests, system test plans are composed by the business team of the client in order to assure the accomplishment of system expectations. Verification can move to the next stage when functional and non-functional requirements are validated.
- User Acceptance Testing: In order to ensure the systems meet the initial requirements, this last stage includes verification plans performed by business users in a real environment, and using realistic data. Lastly, those results suggest if the system is ready for delivery.

Both verification and validation phases are intuitive and well defined, easing considerably the implementation of V-Model. Moreover, testing activities are planned before the actual development, leading to a significant reduction on the validation periods. Thus, a V shaped model has been an effective approach on small to medium sized projects where requirements are well defined and fixed.

2.1.3 MODERN CARS DEMAND MODERN PROCESSES

For many years, traditional models as Waterfall and V-Model successfully filled the software engineering needs. The major advantages are centered on simplicity, well-defined stages, complete documentation, and clear understanding of the project goals. Those characteristics have assuredly brought great benefits to automotive world and software industry in general.

Nevertheless, resembling any other area, software has evolved along the past decades, and became an essential part of the ultimate technology. Thus, same methods that used to support software development are now facing several challenges. Rigorous plans and sequential structures of traditional methodologies such as Waterfall and V-Model turned into serious disadvantages. Due to the rigidity of the models, requirements must be stated explicitly before development, and no working software is produced until late stages of the process. Any change in development objectives might be disastrous since it would mean the resumption of all project. Moreover, it is difficult to measure progress through stages, making the management harder when projects are complex and extensive. Those issues clearly increase the uncertainty and risk, making the projects more susceptible to fail.

Analysing the overall state of global IT projects, the results are far away from achieving the desired success. A recognizable database containing near 50,000 development projects of real-life IT environments [19] shows that although the number of successful projects has been increasing during the last years, the success rates remain far low. The amount of challenged or even failed projects is surprisingly high, as presented on the following chart.

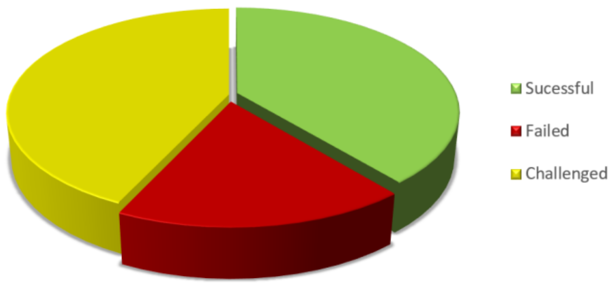


Figure 6 - Project resolution by CHAOS

Only 39% of those projects were delivered on time, with the predicted budget, and comprising the required features and functions. On other hand, 43% were challenged due to late delivery, over budget, and less functionalities. Finally, 18% of those projects failed because of impossibility of completion, or they become useless for the company. These alarming statistics depicted on Figure 6 show the urgent need of new approaches to software industry. Although the use of traditional methodologies is not the problem itself, some practices do not fit the current market and user demands.

As a result of the considerable increase of software complexity, the development of a new product is nowadays an extensive process. Due to its unpredictability, establishing a detailed plan about the development course is becoming more and more difficult. Moreover, the present competitive market often instigates changes on client requirements, invalidating the initial development plan, which has to be reformulated or even restarted. Even when the plan prevails until the end of the project, studies suggest that only 20% of the required features are often used and 50% are hardly never used. Consequently, 20% of the product represents about 80% of the value to the customer [19]. If that valuable part of the product is developed and delivered first in the project, customer satisfaction is rapidly achieved and the time to market might be significantly reduced.

The software industry is aware of those challenges and already concluded that following a rigid plan might not be the best approach to develop every product. That is why new iterative and flexible methodologies such as Agile have been created.

As presented on the opening chapter, Automotive industry is facing the exact same challenges: software complexity has been increasing and has now a central role on automotive innovation, customer demands over the next few years are expected to change like never before, and there is an urgent need to reduce the time to market [20]. Moreover, while the amount of software increases, quality demands for reliability, safety and performance must remain high [4]. The key to overcome these challenges might be on the software development processes. So far the processes in the car industry are not adapted to engineering necessities. Therefore, it is critical to study and experiment new development approaches to support automotive industry.

2.2 AN EMERGING SOLUTION

Simplicity and rigorousness of sequential development processes attest their suitability for projects with clear and well defined roadmaps. On the other hand, when requirements are unknown or subject to change, employing traditional methodologies might not be the best approach.

Software industry became aware of those weaknesses, and progressively instigated the search for new development methods. The main focus always been to conceive a process capable of ensuring flexibility on requirements and a continuous product review. It is not a surprise that, as presented on Figure 7, iterative and incremental methodologies are nearly as ancient as sequential models like waterfall.

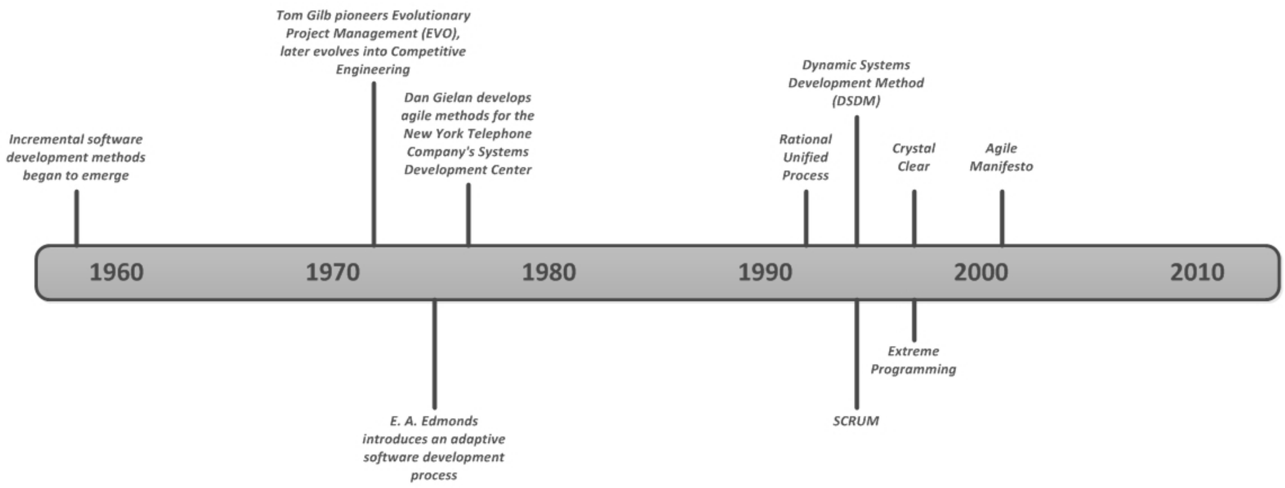

Tom Gilb pioneers Evolutionary Project Management (EVO), later evolves into Competitive Engineering
Dan Gielan develops agile methods for the New York Telephone Company's Systems Development Center
Incremental software development methods began to emerge
Dynamic Systems Development Method (DSDM)
Rational Unified Process
Crystal Clear
Agile Manifesto
1960
1970
1980
1990
2000
2010
Extreme Programming
SCRUM
E. A. Edmonds introduces an adaptive software development process

Figure 7 - Methodologies Timeline [80]

Since its first appearance during the '60's, the concept behind iterative and incremental methods evolved considerably, until 2001 when Agile was formally presented. Agile methodology intends to follow the project throughout the entire development lifecycle. In order to achieve it, development is organized in regular cadences of work, known as iterations or sprints, in which result a potentially shippable product increment.

Instead of an extensive analysis phase, in this "inspect-adapt" approach [21] requirements are gathered continuously along with development. Moreover, since working cycles have fixed and limited periods, stakeholders have recurring opportunities to readjust the product roadmap according to the changing market. Besides reducing both development costs and time to market, this flexible and iterative methodology aims to optimize return of investment and increase marketplace competitiveness.

2.2.1 State of practice

Software industry is aware of Agile potential benefits, and this becomes evident on its increasing utilization. Recognizable companies such as Yahoo, Microsoft, Google, Motorola, SAP, Cisco and many others are already using it [22]. In fact, they are only a small fraction of the growing number of organizations that discarded traditional methodologies in order to adopt an Agile based approach, as depicted in Figure 8.

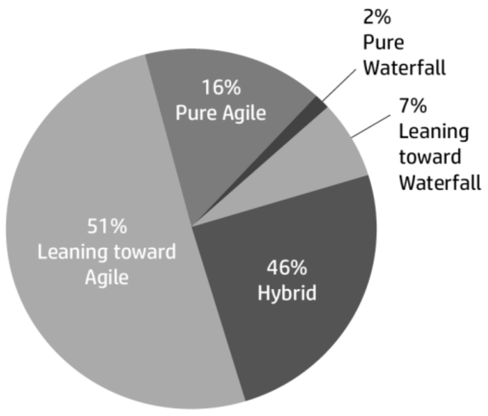


Figure 8 - HP Survey: Development methods usage [10]

Despite being based on a limited number of companies, approximately 600, the survey conducted by HP [10] evidences an increasing mastery of Agile methods. Although the framework has been presented during the 2000's, adoption of Agile practice occurred mostly over the past years. The growth seems to follow the characteristic spreading of innovation into a marketplace.

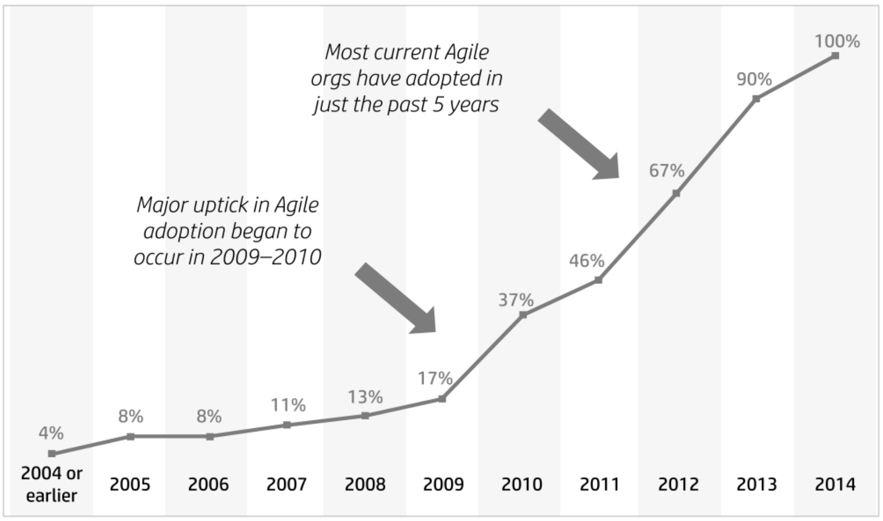


Figure 9 - HP Survey: Agile adoption over time [10]

After the incremental growth, Figure 9 shows a significant inflection point during 2009 and 2010, followed by a sharp increase until recent years. Therefore, Agile is definitely an upcoming methodology, and is currently being experimented across several domains.

Although most of the companies who have adopted a new approach are related to software industry, other sectors such as Financial, Healthcare or even Transportation are experiencing it as well. A survey conducted by a company named VersionOne [23], one of the ancient defenders of Agile, shows the variety of industries which are currently employing it. Distribution of the companies amongst nearly four thousand responses is presented on Figure 10.

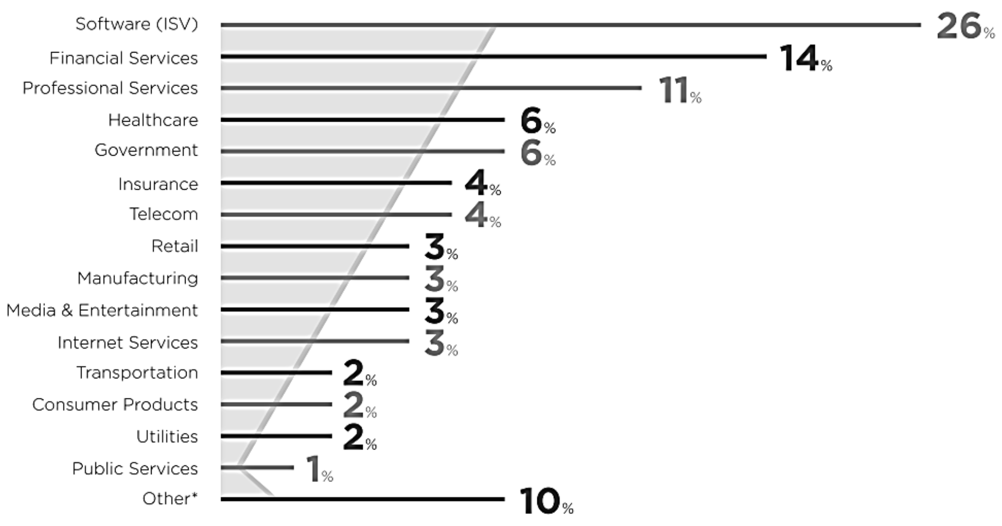


Figure 10 - VersionOne survey: Respondent Demographics [53]

Statistics demonstrate that the Agile methods are spreading by a growing number of industries, proven by the sharp growth in the companies that use them.

However, a decision involving the approach of project management cannot be based on a trend. In other words, companies must somehow benefit from the migration of well-known methods as Waterfall to a relatively new and untested approaches like Agile.

VersionOne's survey shows the major advantages of Agile, pointed by several companies that are already using it. Respondents were able to do multiple choices, presented by order of selection on the Figure 11.

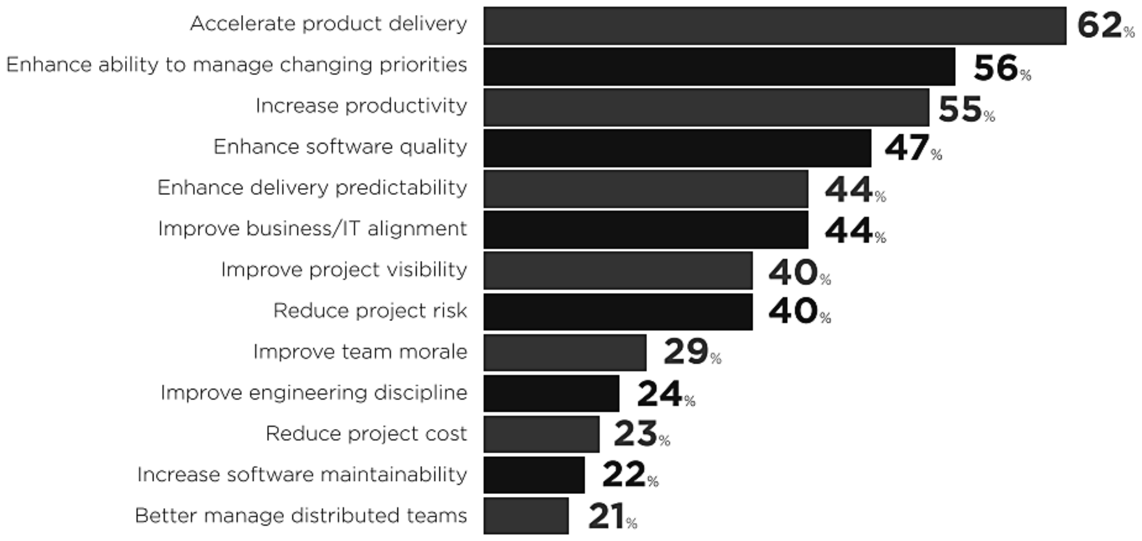


Figure 11 - VersionOne survey: Agile benefits [53]

Agile is clearly in an excellent position after these findings, since are presented great advantages which are assets to any project.

Accelerating the product delivery is presented as the highest rated benefit, and definitely represents a concern when seeking competitiveness and costs reduction. A shorter release period may be achieved, partly due to an increased productivity, which represents another maximum profit. However, faster production is not an advantage if the delivered product is poor. In this sense, Agile maintains the value since one of the major benefits is the enhancement of software quality.

Consequently, there are several reasons to consider Agile as a beneficial approach. Certainty is provided by numerous companies that are already using it, and are keen to point out the advantages of taking that decision.

2.2.2 Vodafone UK: Halving lifecycle through agile

As a leading telecommunications provider, the success of Vodafone UK revolves around innovation and customer satisfaction. In this sense, an online service for existing and potential clients plays a vital role on the business. In order to increase revenues and service levels, software development teams continuously work to add new features to the sites.

Before migration to Agile methodologies, development was conducted through a traditional waterfall approach with a lifecycle of 24 weeks comprising preparation, development and testing. Thus, the process was slow, frequently generating poor quality releases. As a result, the return of investment was non-reasonable since a great part of the costs were associated with testing rather than develop new features. Moreover, lack of contributions by stakeholders often led to poor developed features, causing frequent requests to change and consequent late delivery. This process was not efficient or cost effective and the company was further from its objectives.

Management became aware of these drawbacks and rapidly decided to search for a new approach. Through collaboration with a consultancy company, the entire process was reviewed in order to assess whether Agile software development principles would be more appropriate. Consequently, the established waterfall process was replaced for iterative and incremental development with requirements and solutions evolving via collaboration between customer and self-organized teams.

After an initial period to evaluate the impact of these dramatic changes, results started to show the benefits of the new methodology. The end-to-end process from starting work to release has fallen from 24 weeks to 11 weeks, which has reflected on time-to-market. If the process length reduced to less than half, the overall costs were realigned too. Test and deployment costs went to 20.5 per cent of total costs instead of the previous 51 per cent. Since production issues have fallen, the period of production support has also lowered, conceding staff additional time to conduct more development work. Thus, more functionalities could be developed with a higher velocity, improving client satisfaction. Furthermore, successful and frequent releases motivated the team and led to continuous improvement.

Vodafone UK clearly benefited with the adoption of Agile. After conducting its transformation, the former head of e-technology concluded “We have successfully managed to resolve the change and culture issues associated with introducing an agile methodology to software development. This rapid transformation has quickly delivered effective results to the business” [9].

2.2.3 Samsung: Moving to agile to shorten development

When the subject is technology and innovation, Samsung is assuredly one of the major topics. As one of the most influent brands nowadays, Samsung has decades of experience developing worldwide products which frequently take the lead in the market.

Gadgets and smartphones have been one of the main focus lately, and the launch of new products or upgraded versions is regular. *Galaxy* line of smartphones is particularly well known, as it is one of the most acclaimed categories in the market. For its production, Samsung used to conduct the development through traditional methodologies such as Waterfall. However, despite the veteran development and proved success, the company continued to seek improvement and decided to move towards a more Agile approach.

The results were immediately observed since the development time of the S7 model was expected to shorten by one or two months [24]. Agile key practices such as frequent collaboration, introduction of continuous testing and shorter cycles increased development velocity.

Speeding the development means launch ahead of the schedule, which represents a major competitive advantage in the market. Samsung successfully achieved it by taking a step forward into Agile methodologies. Changing the mindset may be a major challenge, but can also yield major benefits companywide, and the experience conducted under development of last Galaxy model is a proof of that.

2.2.4 Primavera: A Successful Transition to Agile

Primavera is a worldwide software development company, focused on providing enterprise project management solutions that help customers manage their projects, programs, and resources. Despite the experience on supporting other companies developing new products, Primavera has its own struggles. Because of its roots in construction and engineering, Primavera culture used to support the commonly used waterfall development approach. For some years, this sequential cycle empowered the business and positively answered to the changing market demands.

However, it often resulted in working overtime and burned weekends in an attempt to finish projects on time. This routine instigated frustrations and disappointment, especially when the releases were not appreciated by the customers. Furthermore, since the decisions were based on a command-and-control philosophy, the relationships between the development team and other departments began to deteriorate because expectations were seldom fulfilled. Primavera needed to change. Needed flexibility, quick adaptation to market needs, and more involvement from people.

Willing about the risks, the company decided to embrace a new iterative approach, and started to get some ideas from Scrum[1]. Rather than a sudden reformulation, management decided to progressively present and train the developers with the framework. After an initial apprehensive period, the positive results were noticed. Considering the number of customer-reported defects, quality increased about 30 percent in the first nine months [25]. Least time fixing bugs meant more time to development, leading to a four months earlier release.

The true benefits of adopting Agile went beyond measuring features completed and release cycles, because the development team also felt noticeable benefits. Since a sustainable pace was maintained during the entire development cycle, the team was happy and focused on achieving the established goals. This teamwork made the work environment more enjoyable for developers and helped to build trust between them.

Among all the benefits, the major lesson for Primavera was that building software is a continuous learning process, and sometimes changes are needed. Thus, it is not surprising that, after its preliminary implementation in 2004, Primavera has been extending this Agile mindset along their development centres around the world. More recently in 2014, Portugal followed that move when the local organization was entirely restructured in order to embrace a new Agile approach [26]. Two years later, the process was positively assessed with a recognizable certification, attesting its effectiveness.

Still, these good reports should not hide the hard work behind them. After a long experience on software development, changing directions takes time and requires a great effort. Nevertheless, Primavera clearly proved its benefits, and became a model for other companies which are looking to adopt Agile.

[1] Agile approach, described along the following topics

2.3 Agile Software Development

Flexible and iterative methods have been applied over the last decades in order to overcome the shortcomings of software development, spreading and increasing the popularity of Agile methodologies. As a basis for its further implementation, following topics provide a detailed explanation of Agile practices and main approaches.

2.3.1 Background

Agility can be defined as the ability to both create and respond to change, always seeking the profit in a turbulent business environment [27]. Following these values, Agile methods stress two main concepts: the effectiveness of people working together and the honesty of working code [28].

Effective team work enables flexibility, speed, and cost savings. Ideas may be transferred faster when talking directly than through documentation. Moreover, open communication between development team and customers provides opportunities to adjust priorities, identify new difficulties, and discuss alternative paths. In the other hand, working code represents a warranty for developers and customers. Instead of promises or expectations regarding the final product, working code can be shipped, modified or scrapped, but it is always real.

In recognition of these ideas, in February of 2001, seventeen practitioners of several programming methodologies joined together at a summit in Utah, in order to discuss how lightweight software development could fulfil the problems of existing methodologies. Through that work resulted 'The Agile Manifesto' [29] which states four main values for software development:

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

With these values, the Agile Manifesto clearly states what is more important for a better software development. The purpose is not to question the usefulness of processes, tools, documentation, contracts and plans, but to focus and enhance the importance of those four main values.

Relying on interactions between individuals facilitates sharing information and allow a quicker intervention in the process if needed.

Documentation is a useful part of software development process, since it helps to visualize concepts, specify requirements and observe measurements. However, instead of a heavy documentation, working software provides an actual measurement of the project status and enables rapid feedback of the product development.

Customer collaboration means that all members - including the customer, sponsor, developer and user – are on the same team. Thus, merging different experiences and expertise enriches the process and allows to produce more appropriate results. Contracts or project charters are definitely needed, but without continuous collaboration, they may be insufficient.

Working through detailed plans pushes the team to focus on the project and its contingencies. Nevertheless, constant change of requirements, progression of information systems, and new business forms often make obsolete the initial plan. Therefore, rather than focus rigorously on the plan, it is important to flexibly respond to changing realities according to the needs of the customer. These values clearly expose what Agile is about. The main focus is not on the employed practices, but their recognition of people as the primary drivers of project success, coupled with the pursuit for effectiveness and adaptability.

2.3.2 Principles and General Practices

Creativity and autonomy are praised qualities and a key element to succeed [30]. Therefore, an organization is viewed as a complex adaptive system in which individuals interact in self organizing ways, guided by a set of simple and general rules.

Traditional methodologies often provide inclusive rules – all the procedures to be executed under all different scenarios. Teams that follow inclusive rules usually depend on a leader to advance the practices and conditions for every situation. On the other hand, Agile methodologies offer generative rules – minimum set of principles and applicable under all situations to generate appropriate practices. Thus, instead of voluminous written rules, problems are solved through individuals and their creativity.

Based on its main values, Agile defines a set of guiding concepts that should support the entire development process. They are organized in twelve principles[2] involving customer satisfaction, efficiency, iterability, collaboration, product quality, and continuous improvement. These overall rules are common to all approaches, and provide the basis for the general practices that characterize Agile methodologies. The following topics shall provide an overview of the most relevant ones.

- Short Iterations

  Collaboration and responding to change are essential values of Agile methodologies [29]. They are strongly linked, since interaction between customers, development and management is fundamental to face sudden changes on project roadmap. In order to achieve such collaboration and flexibility, the feedback loop must be regular enough to continuously get input from all participants.

  Therefore, Agile approaches recommend to organize development in short iterations, usually periods from two to six weeks, where the team makes constant trade-off decisions and adjusts to new information. These short iterative cycles are combined with feature planning and dynamic prioritization. It means that at the end of an iteration, the customer should be able reprioritize the features desired in the next cycle, discarding originally planned functionalities and adding new ones. This close collaboration makes the product development robust, since its requirements are continuously gathered and readjusted. Consequently, the team has a better understanding of what is desired by the customer, who follows closely the development process. With such a strong cooperation, Agile aims to build confident teams, happy users and satisfied customers.

- Cross Functional and Self-Organizing Teams

  Individual competency is crucial for success of any development process. In other hand, when those competencies are unified in a team that interacts and works together, individual talents are likely to grow, improving the potential of the whole team. Thus, Agile encourages cross-functional teams, where the group should be composed by people having differing personalities and from different functional areas, such as developers, designers, testers, etc. Besides fostering a spirit of cooperation, bringing diversified people together usually improves problem solving, facilitates task

[2] Appendix I – Principles Behind Agile Manifesto

switching, and leads to better decisions. If less time is spent on those issues, project objectives may be achieved earlier, decreasing the production cycle time.

As indicated through 'Agile Manifesto' previously addressed, Agile mindset promotes leadership-collaboration rather than command-control management. In order to achieve effectiveness in such an open process, Agile clearly relies on people and their creativity [31] as part of a self-organizing team. Rather than a leaderless team, this mindset promotes a team that can organize itself in various configurations and scenarios to meet challenges as they arise. Therefore, significant authority and responsibility for many aspects of the work must be given to the team as planning, scheduling, assigning tasks, and take some decisions. Such autonomy stimulates participation, involvement, creativity, and leads to higher productivity.

Therefore, Agile organizational strategy aims to get the best of the team: cross-functionality seeks for individual skills that contribute for team competency; self-organization promotes autonomy, creativity, and responsibility for work.

- Collective Ownership

An Agile team is much more than just agile. Is not only about iterating and responding to change, because "That is what agile teams do, not what they are" [32]. The great Agile teams must encourage a certain feel of professionalism, pride in the work, and above all, an intense collaboration.

This last mindset is, perhaps, the hardest to achieve, because most companies and its developers opt to divide the work into modules or independent sections. They are contributing for the same product and the code will certainly be integrated, but that is not collaborative. Collaboration involves working together in all stages: creating, critiquing, and refining. Rather than claiming ownership over one component of the system, everyone shares responsibility for its overall quality. With this joint commitment to produce good code, every team member shall feel the necessity to discuss designs, explore problems, fix bugs and improve the solution of the whole product [33]. Moreover, when the team is aware of the entire product and aims to improve it together, the risks of concentrating the knowledge in a few members are reduced, assuring the stability of the development if some elements leave.

Since teams usually have individuals with specific skills, taking ownership of unfamiliar code may be challenging. However, having stronger areas of knowledge and intervention shall not be a constraint to learn and embrace different tasks. In that sense, an Agile practice named Pair Programming might be a useful help to get into unknown code. It is a development practice in which two programmers work side-by-side at one computer, continuously collaborating on the same design, algorithm, code or test [34][35]. Besides sharing knowledge, Pair Programming helps the team to learn the strengths and weaknesses of each member, and based on that better adjust the development strategy. Despite having a development-time cost of about 15%, studies suggest that Pair Programming increases quality, reduces defects, improves team communications, and enhances technical skills at significant levels [36].

Nevertheless, the major benefits are not on the practices, but on the collective mindset. By working as a group on a single codebase, is promoted a sense of shared responsibility, shared success, and joint pride of ownership. And usually, it is when individuals learn to think and work together that great teams emerge.

- Continuous Integration

Traditional methodologies rarely dictate the frequency of integration of new source code into the project. Consequently, programmers often work for long periods on their own code without realizing how many conflicts are being generated. Since Agile teams must deliver robust code at the end of each iteration, integrate all work at the end of the cycle would be a long process with a high risk of failure.

In order to prevent those problems, Agile praises continuous integration (CI): “a software development practice where members of a team integrate their work frequently” [37]. Through clean and periodic builds of the system, CI aims to minimize the effort required to each integration episode and ensure the existence of a releasable version of the product.

A simplified CI process embraces four main stages, as depicted on Figure 12.

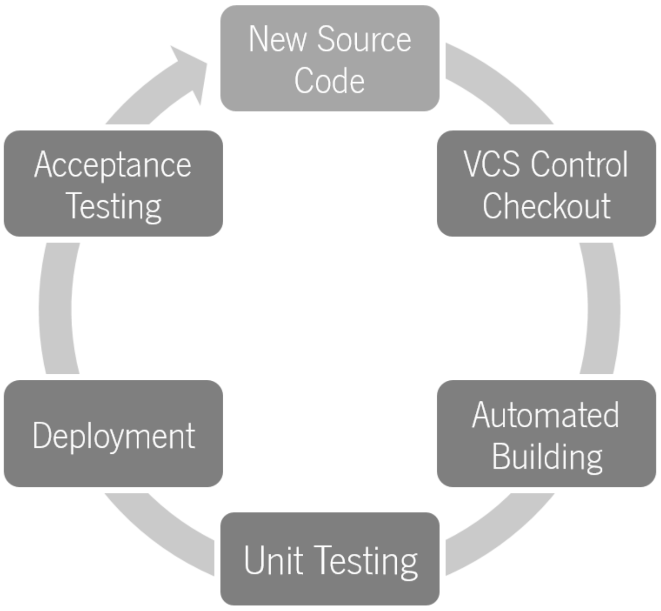


Figure 12 - Continuous Integration stages

Source version control mechanisms became absolutely essential to software development and are no longer an option. In this particular scenario of CI, a new source version of software represents the beginning of the integration process. Typically the Version Control Software (VCS) tool provides detection of a new committed version, which should serve as a trigger to the CI procedure. The first step regards the building of the code, which should be automated. If it does not fail, predefined unit tests can be executed in order to check the integrity of previous developed software. Depending on success of those tests, the new software version may be deployed to the final target, opening the way for acceptance tests. Afterwards, if the process was fully accomplished, the result is a releasable version of the product.

The stages of CI should be adjusted according to the team and project needs. However, the main focus always must be to repeatedly integrate the recent work. This disciplined practice leads to low-defect code with the simplest robust design that fits the features currently implemented. Moreover, avoiding an extensive integration phase definitely speeds up the development time and contributes to earlier delivery of the product.

- Consecutive Testing

Regardless the process, tests always had a key role on software development. Nevertheless, while traditional methodologies consider tests as a phase after development, Agile uses them to guide and support the development. Since it is iterative and incremental, each new portion of functional code can be tested as soon as it is finished.

Figure 13 shows an example of a workflow. New features are added every iteration and must be tested separately in order to finish its development.

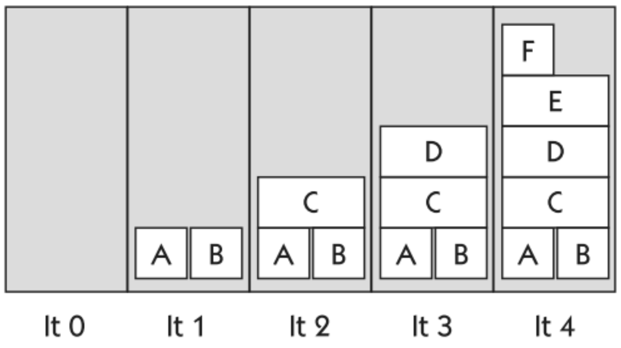


Figure 13 - Testing as part of development [81]

Due to its high importance on validation, tests are part of development, and must be successfully completed in order to finalize the features. Although some teams might have specialized individuals to perform them, Agile organizations are cross-functional, in which several team members may be able to collaborate on tests.

Despite the great importance given to testing, Agile does not specify how it must be implemented. There are numerous approaches such as Unit Testing, Regression Testing, Acceptance Testing, and Test Driven Development (TDD).

Unit tests are short fragments of software, written and maintained by developers, with the main goal of exercise some specific part of the product source code. Through a binary outcome resultant from that evaluation, it is possible to reduce defects in newly developed features, improve software design and allow a better refactoring of the overall code.

In other hand, instead of focusing only the recently developed functionalities, Regression Testing checks the effects of that upgrade on older functions, which apparently are unrelated to the new changes. Thus, besides evaluating the behaviour of the new modification, regression tests also intend to prevent issues on functionalities that have worked properly before.

Acceptance tests are a formal expression of business requirements, since they emphasize functional specifications, frequently derived from use cases or narrative documents. This approach intends to establish a clear and unambiguous contract between developers, customers and users about the product.

Rather than just validation, tests may also drive the entire development process, as promoted by TDD. System requirements are translated into specific Unit Tests, then software is developed or improved only to surpass them, reducing the probabilities of producing useless code. Moreover, design tests before development generally leads to a better structured software, improving the quality of the product.

Despite the testing approach, its automation constitutes one of Agile best practices. Because tests are continuous and not just a phase, running them automatically clearly reduces the effort and time, decreasing the development time.

All of these testing methods and techniques surely make clear that besides seeking a faster delivery, Agile also demands high quality products.

2.3.3 METHODOLOGIES

Rather than predefined rules regarding roles, relationships, and activities, Agile principle-based philosophy allows shaping the methodology to each domain and its necessities. Such flexibility lead to the emergence of several new development approaches under the broad umbrella of Agile, as presented on Figure 14.

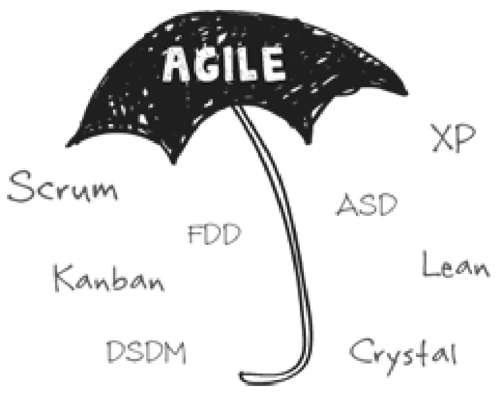


Figure 14 - Agile umbrella [82]

Despite having different strategies, all Agile methodologies follow common principles and even share some practices between them. Following topics shall provide a clear overview of each method.

- XP – eXtreme Programing

After its introduction in 1998 by Kent Beck, XP was described as “a lightweight methodology for small-to-medium-sized teams developing software in the face of vague or rapidly changing requirements” [38]. Success is measured through client satisfaction, which constitutes the main focus of the process. Rather than planning a long term delivery, XP simply emphasizes the current needs. Consequently, customer collaboration during short iterations (usually one to three weeks) is fundamental to discuss and ensure the development of the most valuable features.

Four main values are advocated: communication, simplicity, feedback and courage; and should drive the XP development along its core activities: listening, designing, coding, and testing. The flexibility of these stages is assured through twelve core practices, including planning games, CI, TDD, and pair programming [39].

- Scrum

Aware of the challenges of producing quality work in a changing environment, Ken Schawber introduced a process that “accepts that the development process is unpredictable” [40]. The strategy is on a Rugby practice named Scrum, where the players “huddle closely together ... in an attempt to advance down the playing field” [27]. When playing, each team acts as a whole, as an integrated unit in which every member performs a specific role towards a common objective [41].

That is also true for development teams that embrace the Scrum process. The major focus is on people and on their united effort to achieve a collective success. Thus, individuals work together to iteratively develop the items of a list named product backlog, which contains a prioritized list of all features, functions, enhancements, and bugs. Every sprint (usually with one to four weeks), the highest priority items are planned, moved to an additional backlog, and implemented along the iteration. Finally, a review meeting is held to demonstrate the new functionality to the customer, and create an opportunity for discussion and feedback.

- Lean Software Development (LSD)

After its first introduction on Toyota manufacturing process, Lean principles performed an important role within Japanese automobile industry [42]. The strategy involved reversing the flow of information signals, by pulling materials and components through the production system as needed, rather than pushing them using fully predetermined production plans [43]. This change gave Toyota the ability of make small batches of components "just in time" (JIT), minimizing waste in terms of time and staffing.

Despite being 'born' from the production lines, in the '00's Lean began to be seen as "a synthesis of system of practices, principles, and philosophy for building software systems for a customer's use" [44]. As a result, same core values were afterwards transposed into a software development approach officialised as LSD: an iterative methodology focused on continuous optimization and eliminating waste, which involves effort spend on unnecessary features, partially done work, handovers, defect fixing, and other activities that are valueless to the customer.

- Kanban

Japanese word for "card" or "signboard", Kanban was first used in Lean manufacturing as a scheduling system. Conceived as a flow control mechanism for JIT production, "kanbans" were delivered among the production line as a signal of availability to pull more work. Despite the evolvement of the signalling technology, the system remains at the core of manufacturing today.

In 2004, David Anderson extended the domains of Kanban method, by applying it to a small IT team at Microsoft that was operating poorly [45]. According to Kanban methodology, software development should be driven through three main principles: visualise the workflow, limit work in progress (WIP) at each workflow stage, and measure cycle time [46]. Therefore, the Kanban board plays a fundamental role, since it provides information about the stages of the process, priorities, and current assigned work. Moreover, Kanban visual indicators allow tracking the work in progress. By limiting it to team capacities, a sustainable pace of development is achieved, yielding higher quality products and greater performances [47].

- Adaptive Software Development (ASD)

Originated from rapid application development work by Jim Highsmith and Sam Bayer [48], ASD embodies the principle of incremental and iterative development through constant prototyping. Considering the difficulty in defining requirements for large and complex systems, the strategy is to continuously iterate through three main phases: speculate, collaborate, and learn [49].

Speculate evolves a joint effort with customer to establish goals and plan the development. Then, collaboration to deliver the engineering component and develop the desired features. Finally, at the end of each cycle both customer opinion and technical perspective are discussed in order to improve and adapt planning for the next cycle.

- Feature Driven Development (FDD)

First appeared on a software project at a large Singapore bank in 1997, where Jeff DeLuca managed requirements and its development through an overall model containing a feature list [50]. The process consists in five main activities: develop an overall model, build feature list, plan by feature, design by feature, and build by feature.

In the primary stage, the overall domain model is developed, containing diagrams, classes, relationships, methods and attributes that should express functionality. The object modelled approach constitutes the base for the feature list, which should be planned and prioritized according by the value for the customer. Then, apart from eventual changes on customer preferences, features are designed, built and inspected iteratively until the end of the project.

- Dynamic Systems Development Model (DSDM)

Initially created in 1994, when a large number of project practitioners across many companies joined efforts to build quality into Rapid Application Development (RAD) processes as they developed business-focussed computer solutions. DSDM philosophy states that “best business value emerges when projects are aligned to clear business goals, deliver frequently and involve the collaboration of motivated and empowered people.”[51].

The adoption of Agile practices is by itself a step away from traditional methods, yet the major difference between them and DSDM is on business management. Figure 15 shows that rather than fixing features, DSDM fixes time, cost, and quality.

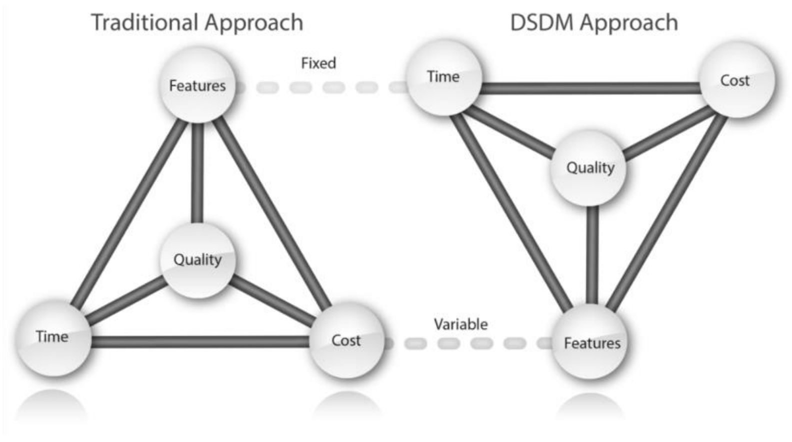


Figure 15 - DSDM Project Variables [51]

Considering that requirements are agreed and duly prioritized, the most valuable features are assured with quality. Moreover, since time and cost are fixed, missed deadlines and over budgets are prevented.

DSDM pursues this reliability through a process of six stages: pre-project, feasibility, foundations, evolutionary development, deployment and post-project. According to DSDM philosophy, the first phase, pre-project, ensures that the business values are aligned and clear objectives are defined. The feasibility phase establishes if those marks appear to be cost-effective, as well as the likelihood of accomplish the project goals from a technical perspective. Then, foundations stage aims to establish a fundamental understanding of the potential solution to the project, and how development and delivery of the solution will be managed. Evolutionary development comprises timeboxes, iterative exploration, MoSCoW[3] prioritisation, testing and other activities that progressively build the solution. Afterwards, the deployment phase consist of assembly, review, and

[3] Prioritisation method with four priority levels: M(Must have), S(Should have), C(Could have), and W(Won't have) [79]

transfer the solution into operational use. Finally, the post-project phase ensures if the expected business benefits have been achieved.

- Crystal

After years of study and interviews of different teams, Alistair Cockburn concluded that following a formal procedure is not mandatory to achieve success. Therefore, those approaches were catalogued into a family of lightweight methodologies named Crystal [52].

When developing software, teams typically have varied skill and talent sets, which should be tailored to project necessities. Hence, Cockburn considered the process as a secondary factor, since the focus shall be on people, interactions, talents, and communications.

Nevertheless, teams of different sizes undeniably need different strategies to solve the upcoming problems. Cockburn aimed to cover that diversity by including several methods into Crystal family, and are divided into colours such as 'clear', 'yellow', 'orange', and 'red'. That indicator denotes the "weight" of which methodology to use in the project: larger and more critical, darker the colour.

2.3.4 SELECTING THE APPROACH

Being Agile has become a trend among the software development industry, and its potential benefits have been tested in several domains with distinct environments. Since each one has its own necessities, it is not a surprising that so many methods have been suggested. Thereby, same core values of a decade ago resulted in a great diversity of Agile approaches today.

According to the report of "Annual State of Agile" [53], Scrum, XP and Kanban are currently the most employed methodologies. Despite being based on the same principles and having common practices, these methods are fairly different. Even among the wide world of Agile, there are different levels of 'agility'. The amount of roles and practices determines how flexible the approach is: less rules, more adaptive; more rules, more prescriptive. Figure 16 presents a scale of the most recognized Agile methodologies, organized according to these characteristics.

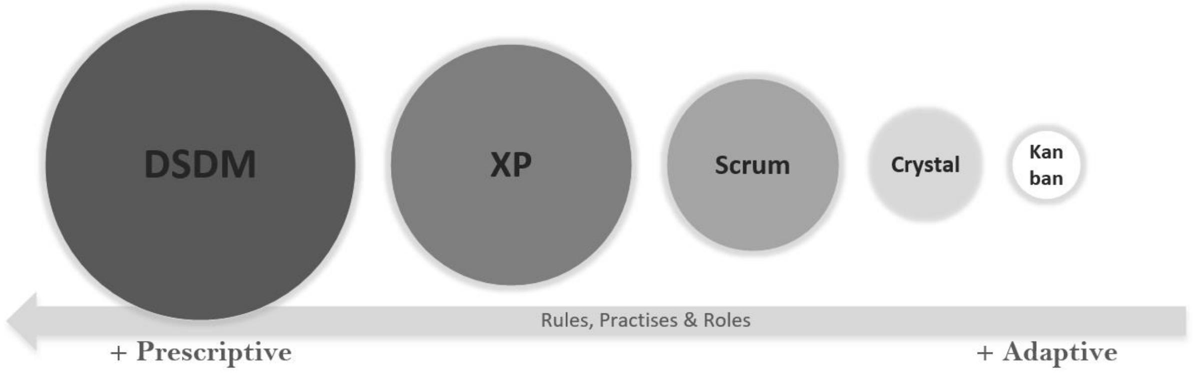


Figure 16 - Agile Methodologies: Prescriptive Vs. Adaptive

Even considering only the major Agile approaches, the previous picture exposes significant differences between them. For example, with a process composed by six stages, nine principles, and more than 12 roles, DSDM is appropriately considered the most prescriptive. In the opposite side, Kanban only promotes six principles, not defining specific practices neither prescribing any roles.

Analysing such contrast does not detract the credibility of any method. Moreover, it proves that every specific environment demands an evaluation of the best approach. Accordingly, the project where this research is inserted already gave that step, by selecting the Agile methodology to implement. Based on the project scope, teams involved, and individuals past experience, Scrum seemed to be the most suitable approach. Thus, following sub-chapter shall provide a detailed explanation on the method and its practices.

2.4 Scrum

As a project management framework that encourages teams to work together and deliver functionalities iteratively, Scrum is unquestionably the most popular of Agile methods nowadays [53]. Based on its official document, the 'Scrum Guide' [40], the following topics shall clarify the details behind such success.

2.4.1 Definition

The Scrum is one set formation in Rugby, where "each team's eight forwards bind together and try to push the opposition eight backwards in order to gain possession" [54]. When playing, a successful Scrum movement requires skill, team work, and a lot of raw power.

Inspired by the game, Ken Schwaber and Jeff Sunderland used Scrum as an analogy to define their development approach: "A framework within which people can address complex adaptive problems, while productively and creatively delivering products of the highest possible value." [40].

As a simple and lightweight process, Scrum aims to manage the development of complex products. Thus, rather than a procedure or a technique, Scrum is a framework where other practices can also be employed.

2.4.2 Principles and Values

The iterability of Scrum has roots on a process control theory, named empiricism. It asserts that knowledge comes from experience and decisions should be based on it. Thus, with an incremental approach, Scrum aims to optimize predictability and continuously improve the process.

As every empirical process, Scrum is founded on three main principles:

- Transparency

  Open communication is encouraged, guided by standards and nomenclatures that should be common to the entire team. Significant aspects of the process must be visible to those responsible for the outcome, including the stakeholders. With their continuous engagement, clients are kept accountable in the development of the product.

- Inspection

  Besides the stakeholders, every member evolved in the process must frequently inspect the artefacts and the progress towards the immediate goals. Without distressing the workflow, such inspection helps to detect undesirable variances.

- Adaption

  When after an inspection some aspects are reported as being deviated outside the acceptable limits, then the process must be adjusted. An immediate correction prevents further deviations, and maintains the development on the right track.

Despite being transcendent to the process, these values are recognisable on specific moments such as deliveries for inspecting, or the start of new iterations to adapt.

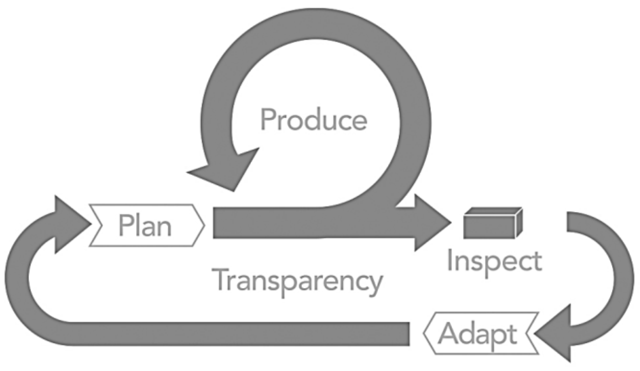


Figure 17 - Scrum Values to Continuous Improvement [83]

When these three empirical principles are duly applied, Scrum becomes way more than just an iterative approach. A transparent process where the team works towards a unique goal, followed by successive cycles of inspection and adaption, which enhance the framework and stimulate its continuous improvement.

2.4.3 ROLES

The team includes everyone who works toward the completion of the product [55]. As an Agile methodology, Scrum teams are cross-functional, so it is critical to ensure that all skills needed to develop the product are covered. The team is typically composed by the scrum master, the product owner, and the core team of developers. An overall scheme is presented on Figure 18.

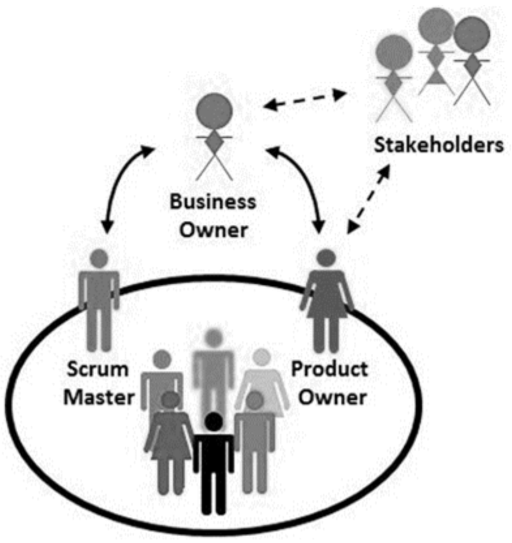


Figure 18 - Scrum Team [84]

The Product Owner is the empowered central point of product leadership. The Scrum Master acts as a coach, facilitator and impediment remover for the team, who is focused on developing the product. This overall picture attests the simplicity of the process and its participants, yet the following topics will provide a detailed explanation on each role.

- Scrum Master

  Responsible for ensuring the framework is understood and enacted. Thus, it must be continuously certified that theory, practices, and rules of Scrum are duly followed by the team. Scrum Master also acts as a servant-leader, by removing potential impediments of the development team and providing help on their interactions to maximize the created value. Moreover, Scrum Master serves the Product Owner in finding techniques for effective product management, implementing them among the development team, and facilitating Scrum events as requested.

- Development Team

  Composed by professionals that work together in order to continuously deliver the requested and committed product increments. The team is empowered and self-organizing, meaning that they are responsible to manage their own work and define the best approach to turn requirements into functionalities. Regardless of particular development domains as testing or business analysis, there are recognized no sub-teams or individual titles than developer. Moreover, despite having specialized skills or areas of focus, accountability belongs to the team as a whole. Such synergy aims to inspire union, and optimize overall efficiency and effectiveness.

- Product Owner

Responsible for maximizing the value of the product and manage its development. Focused on understanding business and customer requirements, the Product Owner prioritizes the work to be accordingly performed by the development team. For that purpose, a Scrum artefact named Product Backlog shall provide information about the more important features to implement, in which the Product Owner is accountable of ordering the items to best achieve the project goals. Moreover, it must be ensured that the Product Backlog is visible, transparent, clear, and contains the information needed to move the development forward. Besides representing a committee, the Product Owner is an individual, whose decisions must be respected by the entire organization in order to make the product successful.

2.4.4 ARTEFACTS

The Scrum Artefacts intend to provide key information for the team and stakeholders about the product status and its development activities. Scrum pillars must be evident throughout the entire process, and here transparency is particularly relevant. Artefacts should be clearly visible to everyone involved, encouraging a common understanding on the information. Such transparency enriches the remaining values, since it enhances the inspection and adaption cycle. That relation shall be clarified through the explanation of each Artefact and its significance on the process, detailed along the following topics.

- Product Backlog

Replacing the traditional requirement specification documents, the Product Backlog is a prioritized list, which contains short descriptions of every functionality to be included on the product. The Product Owner is responsible for its ordering, content, and availability.

Typically, the Scrum backlog comprises items of four different types: features, bugs, technical work, and knowledge acquisition. Features are expressed in the form of User Stories, which are short and simple descriptions of the feature from the user perspective. According to the template *"As a <type of user>, I want <some goal> so that <some reason>"* [56], an example of a User Story applied to web business would be *"As a shopper, I can review the items in my shopping cart before checking out so that I can see what I've already selected."* [57]. Since bugs also express requirements, they follow the same format as the features, and are also inserted into the Product Backlog. Technical work and knowledge acquisition may not be noticeable on the developed product, but they undoubtedly perform a key part on achieving it. To attest their importance, valid examples would be "Upgrade workstations to latest software version" for technical work, and "research and select QML libraries" for knowledge acquisition.

Every item included in the Backlog should have a clear description, order, estimate, and value. Usually the order reflects the value through the position on the list: top items of the Backlog have higher value, and should be developed first. Then, the effort to achieve them is estimated in a collaboration between the Development Team and the Product Owner. Several estimation techniques may be employed, such as Planning Poker, T-Shirt Sizing, or Relative Mass Evaluation [58].

This act of ordering, detailing and estimating the Product Backlog items is entitled refinement. Since just a few features are selected for each sprint, usually higher ordered items are more detailed and better estimated than the lower ones, as depicted through the Figure 19. When the items are duly refined and transparent to the entire team, they are ready to be selected for development.

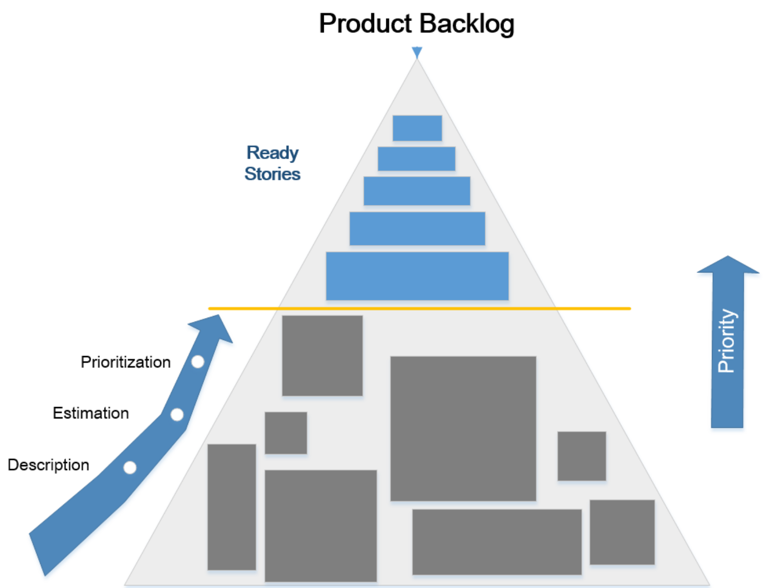


Figure 19 - Product Backlog refinement

Because of the continuous updates, the Product Backlog is never complete. The earliest development is based on the initially known and best understood requirements. Nevertheless, the Product Backlog adapts to market conditions and customer requirements, then accordingly changes to be more useful and competitive. Therefore, the Backlog is a living and dynamic artefact, which reflects the requirements and their value for the product.

- Sprint Backlog

The Sprint Backlog is a detailed list of items committed by the team to be included on the following product increment. It results from a forecast by the Development Team on which items from the Product Backlog can be developed along the iteration. Moreover, the Sprint Backlog should include the work needed to achieve the Sprint Goal, through the decomposition of stories into tasks. An overview of the backlogs and its relation is depicted in Figure 20.

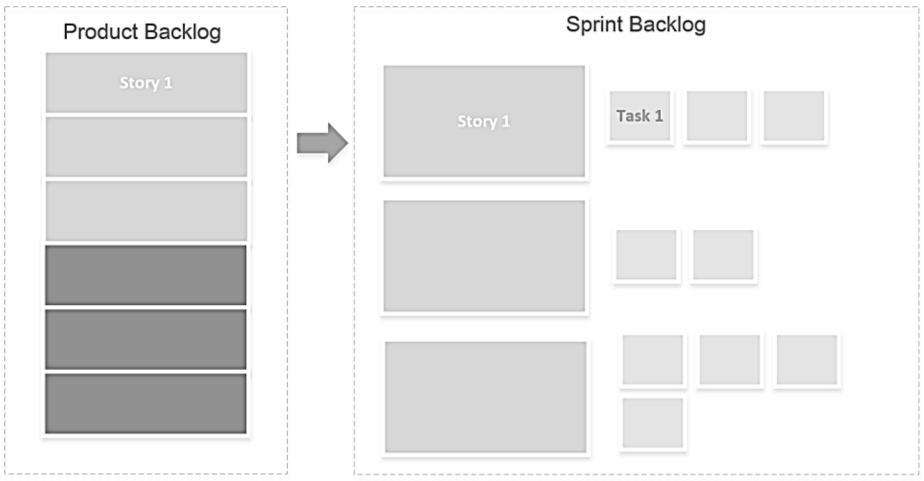


Figure 20 - Sprint Backlog

The Sprint Backlog emerges along the sprint, meaning that it may be completed as the Development Team works and learns more about the effort needed to achieve the committed items. Thus, when new work is required or tasks are finished, the Sprint Backlog should be updated. In order to track and manage the progress, the Sprint Backlog should be highly visible, serving as a real-time picture of the ongoing work.

- Burn-Down Charts

The burn-down charts are a visual indicator of the work progress over a period of time. It is usually presented through a chart that comprises remaining effort depending on time. Effort could be measured in terms of working hours or story points, while time should target the release or the current sprint. An example of a release burndown chart is presented in the Figure 21.

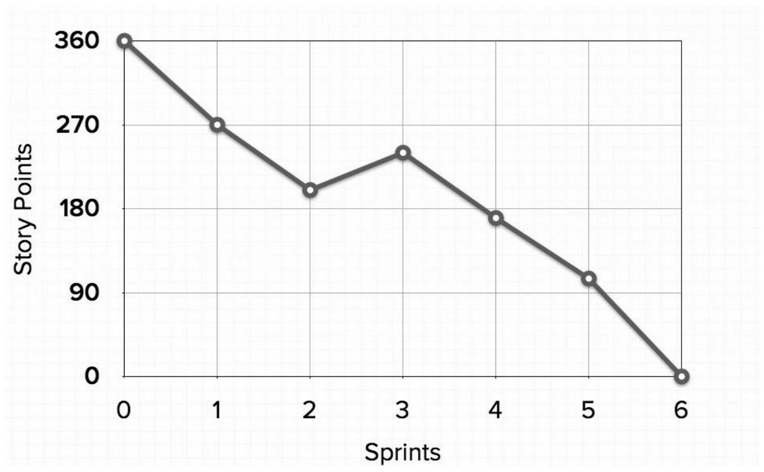


Figure 21 - Release Burndown chart [85]

While some work is progressively completed, other is necessarily added. Considering this volatility, Burndown charts are helpful since they provide a clear vision of the remaining work. Moreover, they enable an efficient tracking of the development pace, which facilitates the estimation on the further iterations. As the responsible for the development, the Product Owner clearly benefits from the measurements provided through this artefact.

- Increment

The increment is delivered at the end of the sprint, and represents the sum of the completed work with the value delivered on previous iterations. It should be shippable, meaning “that all the work that needs to be done for the currently implemented features has been done and technically the product can be shipped” [59]. Nevertheless, in order to be part of the increment, Product Backlog primarily must achieve the ‘Definition of Done’ (DoD).

- Definition of Done

In order to ensure transparency and establish a common standard, the word 'Done' shall have the same meaning for the entire team. DoD is a simple list of activities that add verifiable value to the product [60], such as writing code, comments, tests, documentation, etc. After establishing a criteria, the statement "Feature is Done" is clarified, and becomes common to every team member. Besides enabling transparency on the work status, communication within the team is improved. As the Scrum Team mature, its DoD also tend to expand. Since it constitutes a criteria for delivering the increment, an effective DoD is fundamental to achieve higher quality products.

2.4.5 Events

As an iterative process, time is divided into fixed periods of one to four weeks, called sprints. In each sprint, the team strives to create a potentially shippable product increment, including implementation, tests, and needed documentation. Every sprint must be considered as a project with specific goals, whose quality and content cannot be decreased or changed along the iteration. Thus, in order to continuously improve and achieve the established objectives, it is fundamental to have a well-structured inspect-adapt cycle. Therefore, as presented on Figure 22, each sprint comprises four ceremonies: Sprint Planning, Daily Scrums, Sprint Review, and Sprint Retrospective.

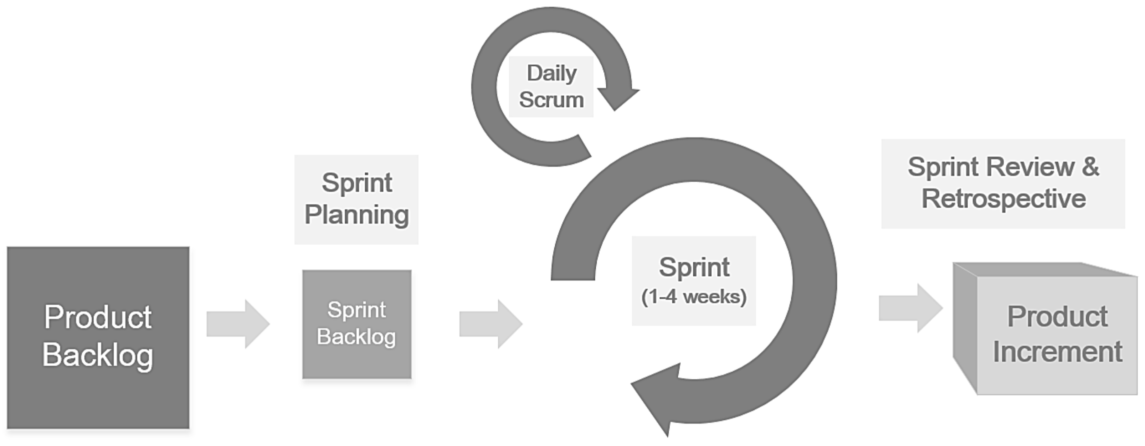


Figure 22 - Scrum Events

- Sprint Planning

Through a collaborative discussion within the entire team, the Sprint Planning intends to define the work to be performed in the current iteration. The meeting is time-boxed to a maximum of eight hours for a one-month sprint, so it should be less for shorter periods. During the planning, two main topics must be addressed: what can be included in the increment for the upcoming sprint, and how will that work be achieved.

Every delivered increment should target the most valuable features for the customer. Consequently, the leading input to define them is clearly the Product Backlog, since it contains an agreed prioritization of the desired functionalities. Therefore, based on past performances and the projected capacity, the development team decides how many top items from the Product Backlog can be accomplished. Then, together with the Product Owner, an overall spring goal is defined. Besides providing guidance along the sprint, keeping this objective in mind should help to visualize the desired increment.

After deciding what work will be done, it is time to define how to achieve it. The development team starts by designing the system and the tasks needed to convert the items on the backlog into a working product increment. The work is planned according with forecast for the upcoming sprint. However, if the work is found to be excessive or insufficient, it may be renegotiated with the Product Owner.

Therefore, the main output of the planning meeting is the sprint backlog. Accordingly, the development team should shall a clear understanding of the established goals and the strategy to achieve them.

- Daily Scrum

A 15-minute time-boxed event dedicated to establish a daily plan, by inspecting the work from the previous day and forecasting the present one. In order to reduce complexity and time, the daily scrum has a fixed schedule and is held at the same place every day.

During the meeting, each member explains what has been done on the day before, what will be done on the present day, and if there are any impediments to achieve it. That regular information is used to inspect the progress towards the sprint goal and the work defined on the sprint backlog. Moreover, it provides an opportunity to improve communications, promote quick decision-making, and to detect eventual problems and miscalculations, which can be immediately addressed.

- Sprint Review

After a cycle of development, the Sprint Review is the occasion to inspect the increment produced along the iteration. Besides being an opportunity for both team and stakeholders discuss the resultant work, the review intends to elicit feedback. Therefore, the meeting should be kept informal, and with a time-box of four hours for one-month sprints.

The Product Owner is responsible for explaining to the stakeholders what Product Backlog items have been successfully achieved, or ‘Done’. Then, the Development Team demonstrates the work, describes how it was performed, and answers questions about it. Finally, the entire group discusses on the following steps. Through collaboration, attendees may realize possible improvements and agree on new opportunities to optimize value. Then, the Product Backlog should be accordingly updated, providing a valuable input for the following sprint.

- Sprint Retrospective

After reviewing the work, the Sprint Retrospective provides an opportunity for the Scrum Team to inspect itself. It is a three-hour time-boxed meeting for one-month sprints, and should occur between the Review and the next Planning.

The purpose of the Retrospective is to examine the flow of the last sprint, regarding relationships, process, and tools. Such collective reflection should lead to a plan for improvements, which must be enacted during the next sprint. Through the application of these improvements, the team completes a cycle of inspection and adaption, which is the essential focus of the Sprint Retrospective.

2.5 Automotive Process Standards

Finding efficient approaches to ensure adherence to universal regulations means competitive advantage. Strategies and practices comprised on international standards constitute a recognized quality indicator. Same occurs within automotive development, which has become an increasingly complex domain. Several standards have been devised in order to attest the safety and reliability of automotive products. Therefore, achieving compliance with these models is vital when designing a development process.

There is a wide range of regulations, models, and standards within the automotive field. The most recognized programs to comply with are ISO 61508, ISO 26262, ISO/IEC 15504 (APICE), and the Capability Maturity Model Integration (CMMI) [61]. While the first two mainly focus safety-related systems, ASPICE and CMMI clearly match the scope of this work, as they emphasize organizations and its development processes. Therefore, following subchapters shall provide a brief explanation on each model and how they can be applied to this project.

2.5.1 CMMI – Capability Maturity Model Integration

"The quality of a system or product is highly influenced by the quality of the process used to develop and maintain it," [62]. This premise stimulated a group of experts from industry and government, along with the Software Engineering Institute (SEI) [63], to create a performance improvement framework and appraisal program [64] named Capability Maturity Model Integration (CMMI). It has assumed great importance nowadays, since several worldwide companies are using it such as Boeing, Intel, NASA, and IBM. In that sense, CMMI has also performed an important role on automotive field, supporting companies as BMW, Bosch, and General Motors [65].

As a product suite, the main goal of CMMI is to provide organizations with the essential elements for effective processes on several fields, including software engineering. In order to support the improvement of organizational processes, CMMI is based on proved practices that have been collected from various organizations and fields of application. Rather than defining a process, CMMI states the core activities to be performed. Thus, focusing what to do instead of the how to do it makes the model applicable to any process and organization.

Achieving compliance with CMMI represents a commitment to reach competitive goals in the global market. Moreover, the formal maturity or capability rating provides an indicator on the effectiveness of the organization and its processes.

- Overview

CMMI is organized in constellations or particular areas, precisely designed to improve a given business need. Consequently, the model focus three specific fields of interest: Acquisition (ACQ), Services (SVC), and Development (DEV). Their purpose and relation is clarified by the Figure 23.

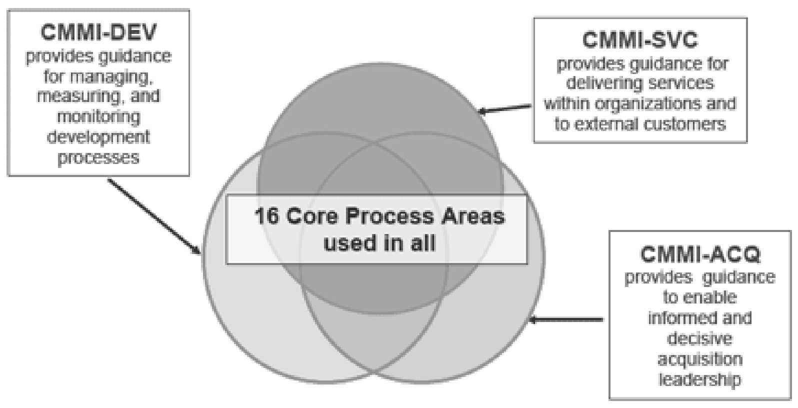

CMMI-DEV
provides guidance for managing, measuring, and monitoring development processes
CMMI-SVC
provides guidance for delivering services within organizations and to external customers
16 Core Process Areas used in all
CMMI-ACQ
provides guidance to enable informed and decisive acquisition leadership

Figure 23 - CMMI Constellations [86]

All constellations have a common strategy to implement several practices regarding project and organizational management, explaining the sixteen core Process Areas. Then, in order to complement these shared activities, each field delineates its specific goals and practices. Due to its relevance for this work, it is appropriate consider CMMI for Development (CMMI-DEV) as the main focus on the following topics.

- Process Areas

A process area is “a cluster of related practices in that area that, when implemented collectively, satisfies a set of goals considered important for making improvement in that area” [62]. Therefore, rather than a single procedure, a process area represents a collection of goals and practices to be achieved, as shown by the following diagram on Figure 24.

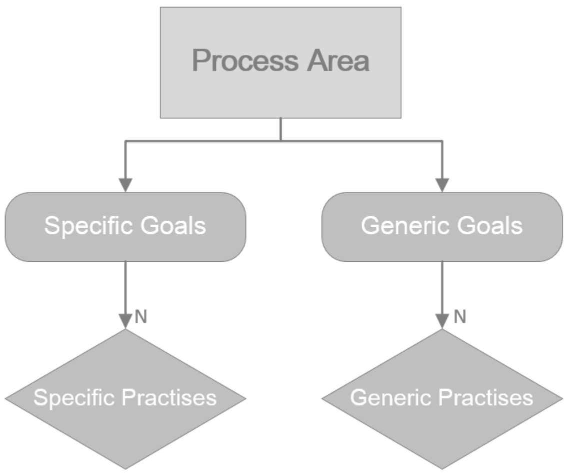


Figure 24 - CMMI Process Area components

Specific goals describe unique characteristics that must be present to fulfil the requirements of the process area, and to determine whether it is satisfied. Accordingly, specific practices define activities that are considered essential to achieve the related specific goal.

Generic goals describe attributes that must be present as a basis for implementing a process area. In order to achieve them, generic practices describe the activities to institutionalize the process, or ensure its consistency within the organization. Both goals and practices are considered generic, in the sense that they are applicable and mutual to several process areas.

CMMI for development comprises twenty-two process areas, in which sixteen are common to all constellations, and the remaining are specific to development. They are divided in four main categories: Engineering, Project Management, Process Management, and Support. A complete list of each category and its process areas presented in the table 1.

Category	Abbr.	Process Area
Engineering	PI	Product Integration
	RD	Requirements Development
	TS	Technical Solution
	VAL	Validation
	VER	Verification
Project Management	IPM	Integrated Project Management
	PMC	Project Monitoring and Control
	PP	Project Planning
	QPM	Quantitative Project Management
	RSKM	Risk Management
	SAM	Supplier Agreement Management
	REQM	Requirements Management
Process Management	OPD	Organizational Process Definition
	OPF	Organizational Process Focus
	OPP	Organizational Process Performance
	OT	Organizational Training
	OPM	Organizational Performance Management
Support	CAR	Causal Analysis and Resolution
	CM	Configuration Management
	DAR	Decision Analysis and Resolution
	MA	Measurement and Analysis
	PPQA	Process and Product Quality Assurance

Table 1 - Process Areas CMMI-DEV v1.3

These process areas perform a vital role on the model, since they define the evolutionary path of the organization and also constitute the basis for its process appraisals.

In order to characterize them, two representations are supported: continuous, where the organization selects the process areas to be assessed; and staged, comprises successive maturity levels with predefined sets of process areas. Following topics shall clarify each model and its measurement levels.

- Continuous Representation

Focuses single process areas for improvement. There are no restrictions on the number or category, so the organization is able to select the process areas to assess and improve. In order to characterize the performance relative to an individual process area, continuous representation uses four Capability Levels:

- *0 – Incomplete:* A process that either is partially performed or not performed. If one or more specific goals are not satisfied then no generic goals need to be achieved, since there is no reason to establish a partially performed process.
- *1 – Performed:* A process that satisfies the specific goals of the process area, and accomplishes the needed work to create products.
- *2 – Managed:* A process whose implementation complies with policy, demands contraction of skilled assets to produce the expected outputs, involves stakeholders, and is monitored according to its purpose.
- *3 – Defined:* A process that is not only managed, but is also planned, tailored, and monitored according to the guidelines of the organization. Moreover, the process experiences even shall contribute to the organizational process resources.

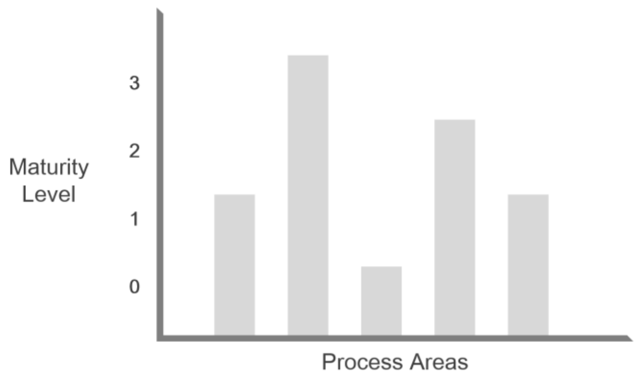


Figure 25 - CMMI Continuous representation

Each capability level comprises a group of generic goals and practices, described in detail on Appendix II. Consequently, a capability level for a process area is only achieved when all generic goals are satisfied up to that level. Such flexibility on evaluating process areas individually provides an opportunity for the organization to focus on their specific needs, improve on particular fields, and monitor the most suitable practices.

- Staged Representation

Rather than considering process areas individually, the staged representation focuses on the organization and its processes as a whole. Stages are composed by a predefined set of process areas, which must be fully performed in order to achieve the corresponded level. To address them, staged representation defines five Maturity Levels:

- *1 – Initial:* Processes are unstable, ad-hoc, and chaotic. Organization occasionally may create working products, yet budget and schedule are frequently exceeded.
- *2 – Managed:* The process is characterized for a specific project. The implementation complies with policy and involves contraction of skilled people with adequate profile to produce the expected outputs. Moreover, it requires collaboration with stakeholders, discipline on monitorization and management, and constant review of the process.
- *3 – Defined:* Processes are based and characterized through organization standards, procedures, tools, and methods. When organizations achieve consistency on their standard processes, upcoming projects tailor new processes based on them. Hence, processes are well defined, managed proactively, and tend to improve over time.
- *4 – Quantitatively Managed:* The organization and its projects establish quantitative goals for quality and performance, which are used as measurement and criteria on project management. Qualitative objectives frequently consider needs of organization, customer, final users, and even people involved in the process.
- *5 – Optimizing:* An organization continuously improve its processes based on past experience and data collected along other projects. Then, considering business and performance objectives, incremental changes are performed involving innovative processes and technological enhancements.

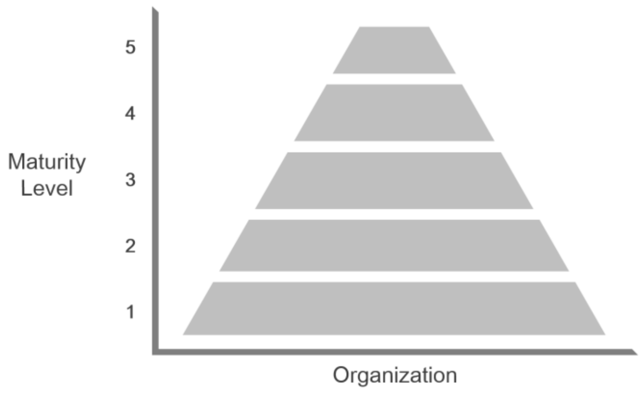


Figure 26 - CMMI Staged representation

Maturity levels enable improvement across multiple process areas in an organization. Predefined sets of process areas must be performed in order to successfully achieve the correspondent maturity level. The relation between each level and its related process areas is explained in detail on Appendix III.

The staged model is an appropriate approach for organizations that seek global improvement. Moreover, achieving a maturity level also enacts as a marketing strategy since it improves the visibility of the organization.

- Assessment Strategies

CMMI delineates appraisals, rather than providing certifications. Appraisals helps the organization to identify strengths and weaknesses on its processes, and to examine how related they are to CMMI best practices. Besides being an opportunity to develop improvement strategies and mitigate risks, appraisals also enable the organization to demonstrate the consistency of its process to customers and business partners [66].

In that sense, the Standard CMMI Appraisal Method for Process Improvement (SCAMPI) is an official model to perform rigorous appraisals and assign quality ratings to organizations [67].

The appraisals follow the same strategies as the representations presented before: assess predefined process areas, and the results are expressed through capability levels; or assess the organization and its processes as a whole, resulting in an overall maturity level.

Regardless the strategy, CMMI appraisals provide guidance for developing and improving processes, and definitely help the organization to focus and achieve its business goals.

2.5.2 ASPICE - Automotive SPICE

The inclusion of complex software and safety-critical systems into vehicles brought new challenges to original equipment manufacturers (OEM's) within the automotive industry. Consequently, the AUTOSIG (Automotive Special Interest Group), including Audi, BMW, Fiat, Ford, Jaguar, VW, and Volvo [68], decided to define a global reference for manufacturers and suppliers named Automotive SPICE. Developed as a variant of ISO/IEC 33004[4] standards, ASPICE represents is a framework dedicated to assess organizational processes related to software and embedded systems development in the automotive industry. It is used either as a status determination for internal process improvement, or to evaluate the process quality of a supplier, acting as a risk assessment tool during the supplier selection [69]. Despite being created in Europe, ASPICE has been expanded to Asia and USA, and nowadays constitutes a prerequisite for becoming a supplier of the most car manufacturers.

The concept of process capability determination of ASPICE is based on a two-dimensional framework presented on Figure 27, consisting of a process dimension and a capability dimension. In order to define them, the model comprises a process reference model (PRM) and a process assessment model (PAM) [70]. While PRM defines the relevant processes to be inspected, the PAM describes how to evaluate its capability within the organization.

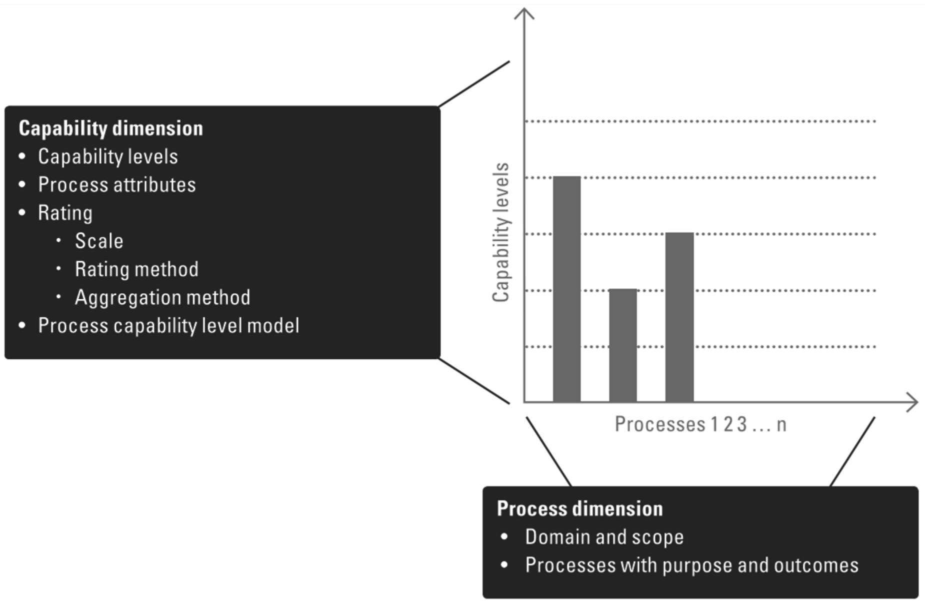


Figure 27 - ASPICE key concept [69]

[4] A revised version of ISO/IEC 15504, also known as SPICE

- Process Dimension

ASPICE defines processes through purpose statements, which define their functional goals when performed in particular environments. Subsequently, each purpose statement has associated a set of specific outcomes, base practices, and a list of expected output work products of the process. The model comprises thirty-one processes, classified into three main categories:

- *Primary Life Cycle Processes*: Embraces acquisition and supply, as system and software engineering process groups, which define requirements elicitation, system design, integration, and qualification procedures.
- *Organizational Life Cycle Processes:* Aim to help the organization to achieve its business goals, through consistent management and improvement practices.
- *Supporting Life Cycle Processes:* May be employed by processes owned by other categories, at determined points in the life cycle.

An overview of each category and its processes is provided in the following table:

Primary Life Cycle Processes	Acquisition Process Group	ACQ.3	Contract Agreement
		ACQ.4	Supplier Monitoring
		ACQ.11	Technical Requirements
		ACQ.12	Legal and Administrative Requirements
		ACQ.13	Project Requirements
		ACQ.14	Request for Proposals
		ACQ.15	Supplier Qualification
	Supply Process Group	SPL.1	Supplier Tendering
		SPL.2	Product Release
	Systems Engineering Process Group	SYS.1	Requirements Elicitation
		SYS.2	System Requirements Analysis
		SYS.3	System Architectural Design
		SYS.4	System Integration and Integration Test
		SYS.5	System Qualification Test
	Software Engineering Process Group	SWE.1	Software Requirements Analysis
		SWE.2	Software Architectural Design
		SWE.3	Software Detailed Design and Unit Construction
		SWE.4	Software Unit Verification
		SWE.5	Software Integration and Integration Test
		SWE.6	Software Qualification Test
Supporting LCP	Supporting Process Group	SUP.1	Quality Assurance
		SUP.2	Verification
		SUP.4	Joint Review
		SUP.7	Documentation
		SUP.8	Configuration Management
		SUP.9	Problem Resolution Management
		SUP.10	Change Request Management
Organizational LCP	Management Process Group	MAN.3	Project Management
		MAN.5	Risk Management
		MAN.6	Measurement
	Process Improvement Process Group	PIM.3	Process Improvement
	Reuse Process Group	REU.2	Reuse Program Management

Table 2 - ASPICE Process domain

- Capability Dimension

Since no predefined groups of processes are imposed by the model, the organization is able to define which processes shall include on assessment. Its Capability is appraised on a scale composed by six progressive stages:

- *Level 0 – Incomplete Process:* The process either is not implemented, or fails to accomplish its goals.
- *Level 1 – Performed Process:* Even without a rigorous plan, the process is implemented and sucessfuly achieves its purpose.
- *Level 2 – Managed Process:* The performed process is properly planned, monitored and ajusted, and its resultant products are controlled and maintained.
- *Level 3 – Established Process:* The process is implemented trough organization standards and base processes, leading to the achievement of its outcomes.
- *Level 4 – Predictable Process:* Process outcomes are meticulously delimited. In order to predict its performance, quantitative management needs are identified, data is gathered and analized to identify possible causes of deviation.
- *Level 5 – Innovating Process:* Data and experience resultant from a predictable process is used to take action, and continuously improve the process in order to better respond to organizational change.

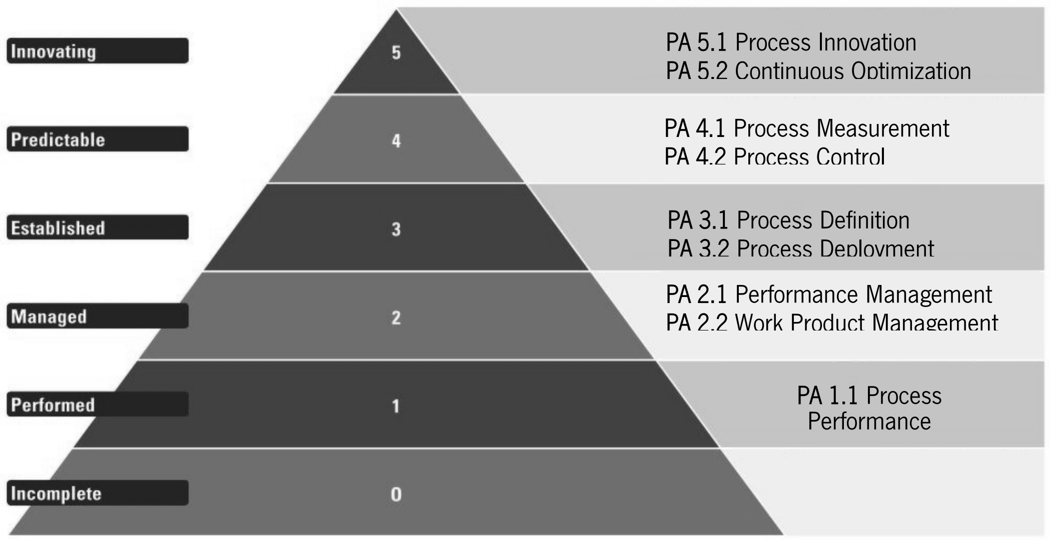

Innovating
5
PA 5.1 Process Innovation
PA 5.2 Continuous Optimization
Predictable
4
PA 4.1 Process Measurement
PA 4.2 Process Control
Established
3
PA 3.1 Process Definition
PA 3.2 Process Deployment
Managed
2
PA 2.1 Performance Management
PA 2.2 Work Product Management
Performed
1
PA 1.1 Process Performance
Incomplete
0

Figure 29 - ASPICE Capability dimension [69] *(Adapted)*

As presented by the picture 29, each specific level is composed by a limited set of process atributes (PA's), wich comprise features apllicabe to the entire process dimension. Therefore, in order to reach a certain capability, they must be achieved by the assessed processes.

In order to measure the extent the achievement of a process atribute and its practices within a specific level, ASPICE defined a four point rating scale comprising performance and capability indicators, both described in detail on Appendix I.

- HIS Group

The increasing importance of software on vehicles represents a constant pressure on manufacturers to extend their competencies and attest the quality of their suppliers. As a result, a group of german automobile manufacturers, including Audi, BMW, Porshe and Volkswagen, created the HIS (*Herstellerinitiative Software*, or 'manufacturer software initiative'). Their common goal is to define and use joint standards for software development.

Since all members were using different approaches for assessing the capability of their software suppliers, a universal strategy based on ASPICE was developed. Consequently, a subset of fifteen processes was selected from ASPICE PRM [70], and named HIS Scope. Engineering processes are clearly the main focus, since they represent the majority of the processes. Table 3 provides a complete list of the processes contemplated by the HIS Scope.

Primary Life Cycle Processes	Acquisition Process Group	ACQ.4	Supplier Monitoring
	System Engineering Process Group	SYS.2	System Requirements Analysis
		SYS.3	System Architectural Design
		SYS.4	System Integration and Integration Test
		SYS.5	System Qualification Test
	Software Engineering Process Group	SWE.1	Software Requirements Analysis
		SWE.2	Software Architectural Design
		SWE.3	Software Detailed Design and Unit Construction
		SWE.4	Software Unit Verification
		SWE.5	Software Integration and Integration Test
		SWE.6	Software Qualification Test
Supporting LCP	Supporting Process Group	SUP.1	Quality Assurance
		SUP.8	Configuration Management
		SUP.9	Problem Resolution Management
		SUP.10	Change Request Management
Organizational LCP	Management Process Group	MAN.3	Project Management

Table 3 - HIS Scope process domain

3. METHODOLOGY

The compliance between Agile processes and automotive development is clearly the main focus of this work. The emphasis on this particular domain arise due to a specific automotive development project, in which the author had a decisive role on design the Agile process and manage its development.

Both scientific research and active participation on a development project created the conditions to apply an action research method. In accordance with its previous description[5], AR intends to solve an immediate practical problem while developing scientific knowledge, and is composed by the stages presented on Figure 31.

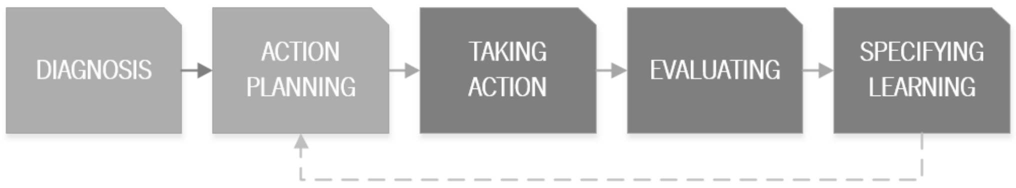


Figure 31 – Action Research Method

Previous sections of the present document already focused the overall challenges of automotive development, as possible strategies to fulfil its current needs. Subsequently, this chapter shall narrow the scope, and focus on the immediate practical problem of AR. The initial topics address the problem diagnosis through a detailed description of the automotive development project. Following, the strategies to implement a suitable Agile process and manage the product development represent the action plan. Finally are described the methods for the evaluation phase, focusing on the process compliance with automotive standards.

[5] Detailed under Research Methodology.

3.1 Project Bosch InnovCar: “Cockpit of Future”

As a co-promotion initiative with a recognizable company within the automotive world, this project constitutes a unique opportunity to acquire more knowledge on this particular domain. An overview on the project structure and its global objectives shall clarify the purposed challenge.

3.1.1 Background

The project “Cockpit of Future” is promoted by a collaborative investigation program between Bosch Car Multimedia [71] and University of Minho. While Bosch offers a strong and clearer vision of the automotive business, UMinho ensures innovation and technological knowledge. The combination of these competencies into a unique initiative intends to originate new concepts and innovative ideas towards the car of the future. This particular project focuses on advanced HMI systems for automotive, in which the modern topic of autonomous driving performs a determinant role. Moreover, the concepts are submitted to usability tests and validated according to specific scenarios in a Driver Simulator Mockup (DSM). Therefore, the project comprises several domains as engineering, ergonomics, human factors, and simulation. Since their intervention is closely related, the development process performs an important role to manage their cooperation and achieve success.

3.1.2 Aims and Goals

Combining new ideas and innovative technologies into a futuristic HMI represents the overall target of this project. However, due to the extension of the scope and the several teams involved, the objectives may be divided in three main areas:

- HMI Concepts and Systems: Considering the different levels of autonomous driving, developed concepts shall ensure the focus on driving task; inform the driver on the current autonomous decisions and provide alerts when intervention is needed; monitor the driver workload and cognitive state including fatigue, stress, distraction, and drowsiness; and adapt HMI systems according with driver profile, past history, and current driving scenario.
- Toolchain and Architecture: As a basis for the development of the HMI systems, a suitable toolchain must be established. It should consider the hardware and software requirements for the target systems, and provide the needed resources to develop them. Moreover, both toolchain and architecture should ensure the interoperability with the systems and platforms

existent at Bosch, in order to enhance the development and ensure compliance with the current automotive standards.

- Usability and User-Experience: Includes the creation of new tests and simulation scenarios in order to validate the developed HMI systems. They shall ensure the compliance with the employed platforms and technologies, and finally be implemented on the DSM (Driving Simulator Mockup).

3.1.3 Structure and Organization

Considering the scope of the project, four different teams may be distinguished: human factors, platforms, simulation, and engineering. Each group has a determinant part on accomplishing the established objectives previously presented. Thus, in order to organize the work and clarify the expected results, the overall project was divided into intervention areas or sub-projects, named Work Packages (WPs). Table 4 presents a complete list of the envisioned WPs.

ID	Work-Package
WP1	Wrong-way Driver Warning concept – WDW
WP2	Strategy on how to address warnings
WP3	Workload management
WP4	Driver monitoring for HMI (incl. eye tracking)
WP5	User Interaction Technologies
WP6	Personalization / HMI adaptation – Intelligent HMI
WP7	Autostereoscopic displays
WP8	3D HMI
WP9	Development of competencies for 3D HMI development
WP10	New usability evaluation methods for ADAS
WP11	HMI for Autonomous Driving
WP12	Multi-Modality
WP13	New HMI development & validation process
WP14	Updated DSM
WP15	Instrumented and integrated vehicle
WP16	New methodology for HMI platform selection
WP17	New HMI platform (HW)
WP18	Requirements capture framework

Table 4 - Bosch Innovcar P689: Project work packages

Since this dissertation addresses primarily the development process, the major focus of this work will be on the engineering team. Nevertheless, the awareness of the wide scope and the several groups involved definitely emphasize its great importance of the project. In order to succeed in such a demanding field and ensure the collaboration with other teams, it is fundamental to design a flexible but consistent development process. Accordingly, the process is addressed by the WP13 and WP18, which respectively discuss the development strategy and the requirements capture framework. Therefore, besides having a direct output for the defined Work Packages, the development process conceived along this work will perform an essential role on the project.

3.2 Development Strategy

Establishing a reliable development process is fundamental for such an ambitious and challenging project. Firstly, it is needed to consider the scope and teams involved, and clarify the overall requirements for the process. They shall provide the basis for defining a suitable development approach, described along this section.

3.2.1 Process Requirements

Guide the teams towards the project goals is the main concern of the process. Therefore, the initial step before defining the process requirements is to organize the teams into an overall workflow, capable of fulfilling the project main objective of creating new automotive HMI concepts. The following diagram in the Figure 32 provides a global structure and organization of the involved groups.

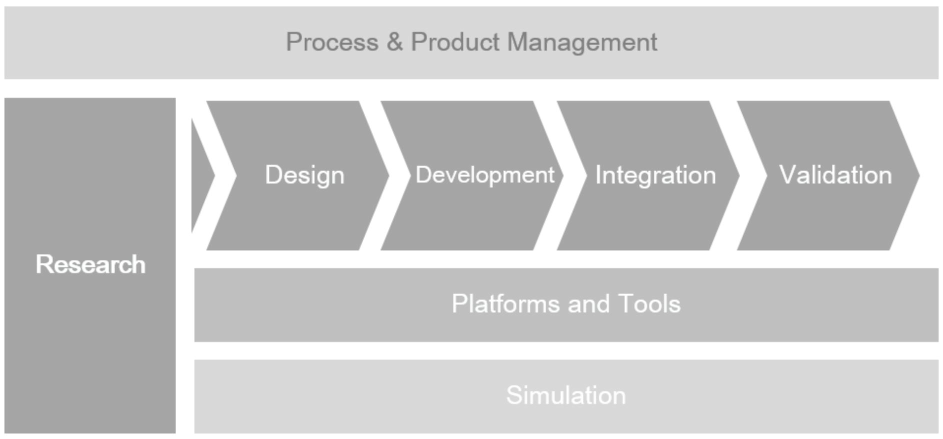


Figure 32 - Overall workflow

Research teams provide inputs for implementing news concepts, whose production comprises design, development, integration, and validation. Additionally, platforms and tools, and simulation teams provide support along the entire process.

Since research will be driven according to the work packages, it is expected to have several investigation lines. They perform a fundamental role on the project, as they constitute the major input of new concepts and technologies into the development workflow. Therefore, the process must ensure constant but structured communication between research and design teams.

When the concept is mature and duly designed, it is developed or implemented on the target platforms, then integrated into the DSM, and finally validated with usability tests. As a sequential workflow, each team can only manage one concept at a time, which afterwards is delivered to the next team on the line. In order to achieve such synchronization, the process must ensure a common and well-structured plan as a constant feedback amongst the teams.

Same characteristic is required regarding the supporting teams. Since the target platforms and operative systems regularly get updates and new versions, it is important to ensure that constant flow of information to the development team. Moreover, a reasonable part of the developed concepts demand interaction with the simulated environment, so the correspondent team must actively participate and discuss strategies with the development team.

These interactions between several teams, combined with the wide scope of the project, reveal the difficulty of establishing a detailed plan. Accordingly, they lead to the final and most important requirement: agility. In order to guide such a complex organization towards the unpredictable priorities of the project, the process must be flexible and transparent to all involved members. Therefore, the process shall be designed according to Agile principles, and shaped to meet the project specific needs and fulfil automotive development standards.

3.2.2 Process Specification

The challenge consists in combining flexibility and robustness into a lightweight framework, which shall support and drive the teams towards the project goals. Amongst the wide range of Agile methodologies, Scrum represents the balance between adaptability and predictability, and clearly provides a suitable solution for the project necessities.

After considering its detailed explanation on the previous chapter, now the framework needs to be structured and settled to this specific case of development. Recalling the basics, Scrum organizes the development into iterations of one to four weeks, called sprints. At the end of a determined number of sprints, an increment of the product is released. Besides the planning, each sprint comprises inspect and adapt events, namely the reviews, retrospectives, and daily scrums.

Since these ceremonies are already delineated on the Scrum Guide [40], the most important decision lies on time-boxing, or defining their length. Thus, considering the needs and structure of the project, sprints have been organized in periods of two weeks. As medium sized sprints, the work is constantly tracked and reviewed without spending excessive time on events. After two iterations follows a week of release, intending to present the results and formulate the overall plan for the next sprints. The diagram in the Figure 33 illustrates the global structure of the delineated process and its events.

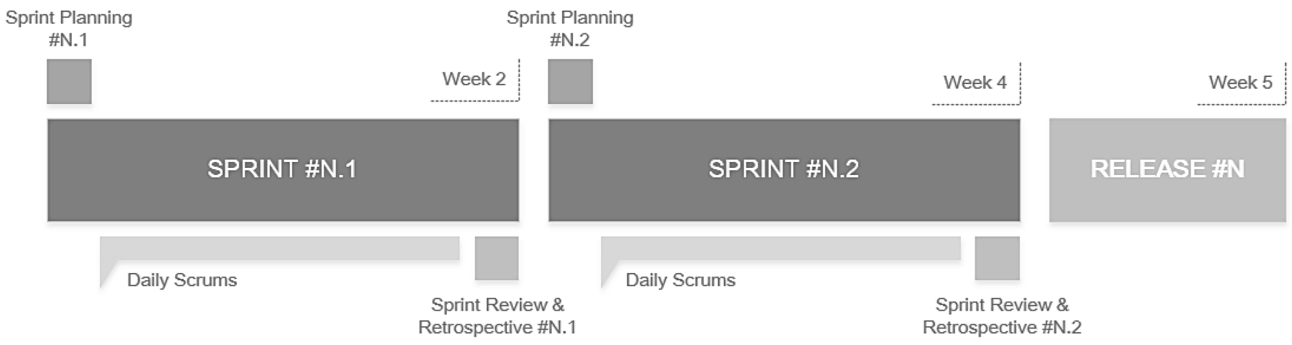


Figure 33 - Scrum: organization and timeboxing

The process flow shall be consistent and coherent with the Scrum rules. Therefore, each sprint comprises the following events:

- Sprint Planning: (Max: 2h00) Takes place at the beginning of every sprint, and intends to plan the upcoming work. Besides defining which features from the product backlog may be implemented during the iteration, the discussion must also focus on the work and tasks required to achieve them. The major output of the meeting is the sprint backlog, which should be accordingly updated and clear to the entire team.

- Daily Scrums: (Max: 0h15) Short and on place meetings to track the development along the sprint. Their purpose is to inspect the work produced on the day before, establish a short term plan for the present day, and report eventual impediments.
- Sprint Review: (Max: 2h00) An opportunity to present results and elicit feedback on iteration outcomes. Thus, this event may include participants outside the development team, as company representatives or even members from other the research groups.
- Sprint Retrospective: (Max: 1h30) Team internally discusses its performance during the finalized sprint, identify weaknesses and drawbacks, and draw future improvements.

After a month of development, corresponding to two sprints, follows the release week. Although the sprint results were already demonstrated on the review meeting, this week constitutes an opportunity to integrate that work, and prepare a formal demonstration to company representatives, project coordination, and other teams involved. As a result, all the participants acquire a transparent view of the product status, and become able to discuss and give feedback on its progress.

Besides being a formal period of inspection, the release week also intends to prepare the next steps on the project. The several teams involved may reunite, present their research advancements, and suggest possible improvements. A collaborative discussion may include new product functionalities as a reprioritization of the already existent from previous brainstorms. After the release week, teams must be aware of the project status, and confident about the next steps.

3.2.3 Supporting Practices and Tools

Scrum events attest the consistency and predictability of the framework, as they create specific occasions to guide and track the development. However, despite organizing the development, Scrum does not address how it should be performed. Teams shall be self-organisable, meaning they are responsible to agree on methods and strategies to achieve the purposed goals. Nevertheless, in order to ensure quality on the delivered work, the development must follow disciplined and consistent procedures. Consequently, some additional practices were added to support the Scrum base framework.

An effective coordination is the first step to achieve a consistent software development process. Team work requires organization, especially when collective code ownership is employed. In order to manage source code and other important artefacts, it is essential to employ a Version Control System (VCS).

Besides providing a clear insight on recent changes, VCSs allow regression to previous versions of the project, enabling team members to safely collaborate on the same files. Amongst the a wide variety of available version control systems, and this particular project will implement GIT [72]. Considering both project and team organization, the tool positively satisfies the requirements. Moreover, the achievement of the highest rate among users proves GIT as a suitable solution [73].

The control of software changes leads to the next practice: continuous integration. Since new code is often a source of conflicts, it would be beneficial to progressively integrate new functionalities instead of a final and long cycle of integration. In this sense, automatic builds may be triggered with specific schedules or simply by a new version submitted to a control mechanism such as GIT. In order to implement that automated behaviour, an integration tool named Jenkins [74] will be employed. Located at a local server mutual to GIT repository, Jenkins triggers a new build every time a developer checks in changes on the source code. Moreover, it presents a dashboard with detailed information on results of previous builds, performance reports, and eventual error messages. An overview of both CI and VCS tools is presented in the Figure 34.

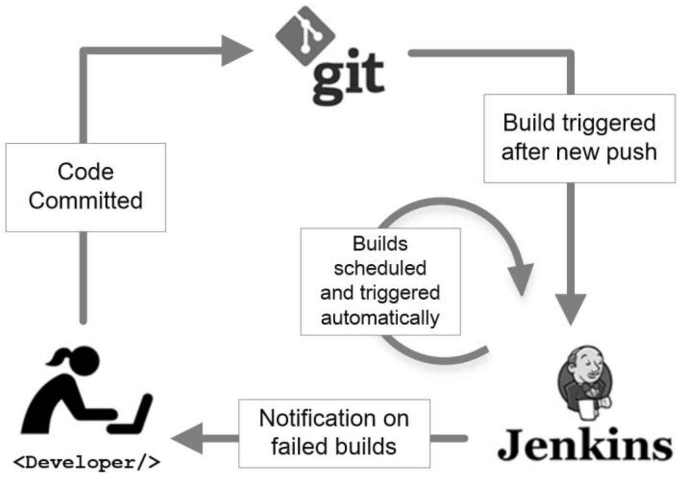

git
Build triggered after new push
Code Committed
Builds scheduled and triggered automatically
Jenkins
Notification on failed builds
<Developer/>

Figure 34 - Continuous Integration system

In this particular continuous integration approach, integration starts whenever new code is committed to local repository, which automatically triggers a build on Jenkins integration tool. Additionally, builds must be scheduled and automatically performed several times a day. Besides providing a graphical view of history and performance, building outputs also attest code stability and reveal eventual errors. In order to ensure safety on this pipeline, the repository and its changes shall be hosted online through a service as GitHub [75] or Bitbucket [76].

Despite ensuring code stability, behaviour and functionality are not evaluated by integration tools. Therefore, the final stage after deployment is to verify the recently added features and attest the conformity with established requirements. Often a result from use cases or narrative documents, these requirements shall be expressed in a form of acceptance tests. As a specification contract between customer and developers, acceptance tests comprise user level features and operational requirements as quality of performance. Once a new functionality and related code surpasses all the tests, is considered done and integrated. Accordingly, the flow presented in the Figure 35 must be achieved.

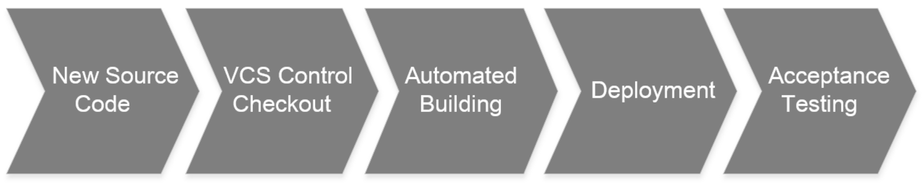


Figure 35 - Development flow

Since the process and its practices must be defined according to both team and project necessities, CI and Acceptance Testing phases were defined according to available tools and the effort needed to implement them. Still, as every Agile approach, the main focus of this development process shall always be on repeatedly integrate and delivery the work.

3.3 PRODUCT MANAGEMENT

An effective management of the product artefacts is essential to a successful project. Agile is clearly aware of that importance, and several methodologies define specific management activities. Scrum in particular, defines Product Owner role. Focused on understanding business and customer requirements, the Product Owner prioritizes the work to be accordingly performed by the development team, "bridging the gap between 'the suits' and 'the techies'" [77]. Such description defines the responsibility of this author on the present project. As a Product Owner, he is responsible for the interface between Bosch Car Multimedia and the development team. Besides managing the project artefacts, additional activities related to requirements gathering and prioritization must be performed, as described along the following topics.

3.3.1 Vision and Plan

Sketching the future product and its desired characteristics is essential for its accomplishment. Such vision acts the overarching goal and guides the entire organization towards its achievement. Thus, it shall communicate the essence of the future product concisely, and describe a unified goal capable of providing direction, but general enough to stimulate creativity.

When starting a new project, it is often difficult to establish a long term plan. Even customers and stakeholders are uncertain on the roadmap; so they are not capable to state the product future characteristics. An eventual solution is to make use of Scrum first releases, and iteratively construct the product vision through simple demos and prototypes. Being an opportunity to free innovation and creativity, this preliminary iterations enable the test of new technologies and architectures, providing a grounded base for the following work. As a result, this joint effort between customers and development team shall incrementally enlighten the product and define its vision.

3.3.2 Requirements Gathering

Requirements are critical for defining, estimating, and managing the development; so gathering them effectively is the cornerstone to a successful project. Traditional approaches define the entire set of requirements on an early phase, which usually becomes a lengthy process. In other hand, requirements under Agile methodologies are collected iteratively and through costumer collaboration. Rather than define every detail on *what* the system must provide, continuous discussion along the iterations intends to clarify *how* the system must work.

Therefore, since requirements primarily target the functionality, Scum express them as short and simple descriptions named 'user stories'. Written from the user or costumer perspective, user stories follow the template: *"As a <type of user>, I want <some goal> so that <some reason>".* One of the major benefits of an approach with user stories is their varying levels of detail. When large amounts of functionality are covered for a single user story, it is named as 'epic'. Due to its extension, an epic is excessively large to be completed in one iteration, so it shall be divided into smaller user stories in order to be modularly implemented.

Despite the detail and granularity of user stories, they might not be enough to describe the intended feature. Consequently, each user story shall be accompanied by an Acceptance Criteria, which according to Microsoft are "conditions that a software product must satisfy to be accepted by a user, customer, or stakeholder" [78]. Accordingly, acceptance criteria comprise a set of statements that specify functional, non-functional, and performance requirements of the product unit. Through a clear pass or fail result, criteria help to determine the boundaries of a story and when it is fully completed. Following diagram provides a practical example on the requirements capture framework to be implemented.

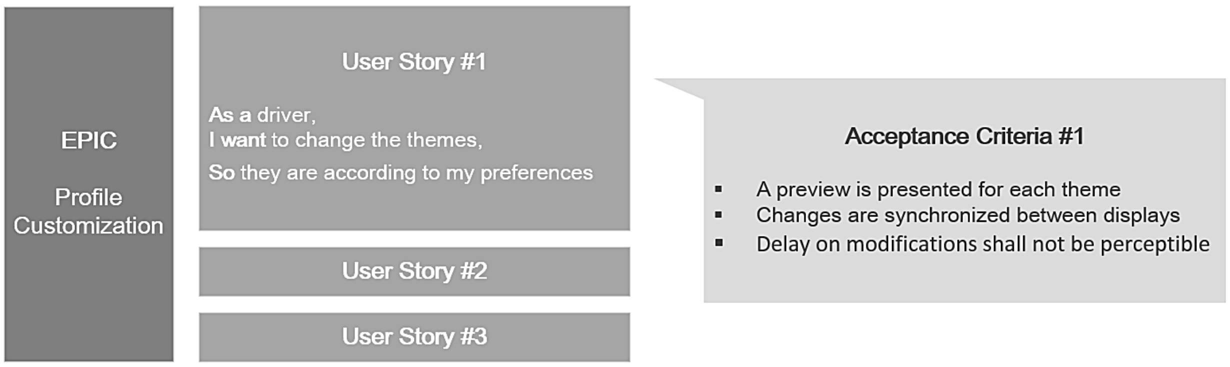

EPIC
Profile Customization
User Story #1
As a driver,
I want to change the themes,
So they are according to my preferences
User Story #2
User Story #3
Acceptance Criteria #1
A preview is presented for each theme
Changes are synchronized between displays
Delay on modifications shall not be perceptible

Figure 36 - Requirements gathering framework

User stories and consequent acceptance criteria shall be defined with a collaborative participation of customers and stakeholders. As a result, the development team acquires a clear understanding on what needs to be accomplished, while the customers obtain realistic expectations on the outcomes.

3.3.3 Development Management

After discussing and gathering requirements, they must be transposed to the development team. The product owner shall provide detailed explanations on user stories and acceptance criteria, so the development team acquires a clear understanding on the next steps. Then, as the product owner defines the short term plan through work prioritization, the team estimations provide a forecast on the effort needed to achieve it. Furthermore, the tracking and monitorization along the sprint help the product owner to evaluate the conformity of development with the established goals. A brief description on this activities is provided by the following topics.

- Product Backlog

Product backlog is a list of features to be included on the product. Being expressed on a user story format, requirements are listed and prioritized according to value for customer. Since they tend to change and evolve over iterations, the product backlog content may be modified on an ongoing basis.

When clear and accurately prioritized, backlog items become ready to be selected for development. Then, the team forecasts the viable amount of work to be conducted under the sprint. As a result, the committed items are transposed to a short-term artefact, the sprint backlog. Despite following the same structure as the product backlog, the sprint backlog adds detail to its items, including description, order, effort, and respective acceptance criteria. In order to comprise that information and present the backlog items, the following template in Figure 37 was established.

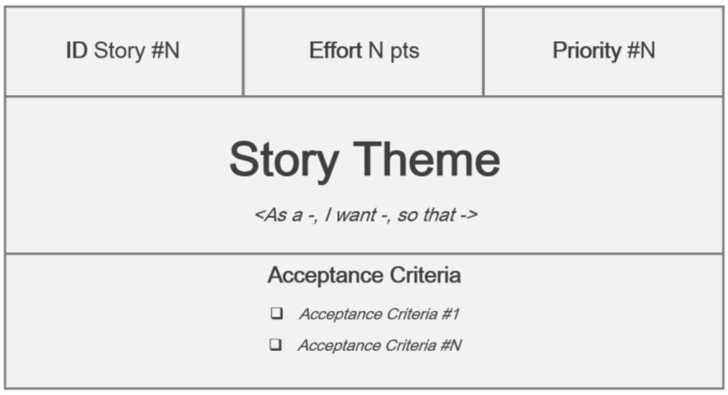


Figure 37 - Sprint backlog item

The product owner has accountability for both product and sprint backlog. More than populating them with user stories, the product owner must prioritize them according to customer requirements, and ensure a clear understanding from the development team.

- Prioritization

The backlog order dictates the upcoming items, their prioritization performs an essential role on the development plan. Prioritization is responsibility of the product owner, who shall consider three factors: value, risk, and dependencies.

An item is valuable if it is indispensable to bring the product to life. Therefore, an item shall be dispensed when product could still achieve the intended benefits without it. As a result, the product will only implement the minimum functionality, without consuming time and energy on unnecessary features. Moreover, customer feedback on new increment will be centered on truly important functionalities.

Risk is an intrinsic part of software development, and it is closely related to uncertainty and lack of knowledge. Lack of knowledge causes uncertainty, which clearly represents a risk for process success. Therefore, uncertain and risky items should have the highest priority. Despite enforcing early failure, this risk-driven approach intends to accelerate the generation of new knowledge and change the course while there is opportunity.

Finally, prioritization must also consider dependencies between functional and non-functional requirements. As they dictate the development flow, dependencies limit the freedom to prioritize the items and estimate their effort. Therefore, dependencies shall be mitigated whenever possible, possibly by achieving modularity through smaller items.

- Estimation

Estimating backlog items provide an impression on their rough size and respective effort. Consequently, estimations enable to forecast and track the work along the sprint. As a joint effort within the team, they are conducted during the sprint planning, or whenever the understanding of an item changes.

In order to perform estimations, the first step is to define a measurement scale. Forecasting development hours is nearly an impossible task, so the approach consists in estimating raw effort and size through a relative measure, named story points. Rated than defining an absolute value, points shall be assigned according to a reference story. Table 5 presents the implemented reference scale.

Story Point	T-Shirt Size	
1	XS	Extra Small
2	S	Small
3	M	Medium
5	L	Large
8	XL	Extra Large
13	XXL	Double extra-large
20	XXXL	Huge

Table 5 - Story points scale

An accurate estimate requires evaluation of possible dependencies. Moreover, the team must have enough knowledge on the needed procedures for its implementation. Otherwise, a preceding item must be added so that relevant knowledge can be acquired. Although estimation is only assigned to team members related to development, the product owner shall be present in order to explain and clarify the items to estimate.

- Progress Tracking

Development conducted along the sprint is continuously inspected through daily scrums. However, besides the scrum board, there is no visual representation on the remaining work. In this sense user story estimates perform an essential role, as they provide a concrete forecast on the ongoing progress. Each sprint starts a burndown chart where story points are presented by elapsed days. When a user story is finished or some considerable part is achieved, the remaining points on the burndown chart are updated. Besides presenting the remaining work, burndown charts also provide concrete data on performance of the team, which may be used for inspection and further retrospectives. Moreover, the number of achieved story points per sprint show the average velocity of the team, constituting a basis for improvement along future iterations.

3.4 Process Appraisal

Achieving compliance with industry standards is essential when designing a development process. Regardless the achieved level and maturity, standards provide a clear insight on process weaknesses and consequent improvements. Moreover, a standardized process undeniably receives more recognition from customers and competitors. Considering all of these benefits, it was decided to use both CMMI and ASPICE as appraisal models. More than an overall rate, these models shall indicate an improvement path towards an effective process for automotive development.

Since CMMI and ASPICE have different structures and appraisal methods, it is needed to define a common strategy to compare the results. Despite having different names, CMMI's Staged Representation and ASPICE's Capability Dimension follow the same structure presented in Figure 38, in which results are expressed in a form of a level.

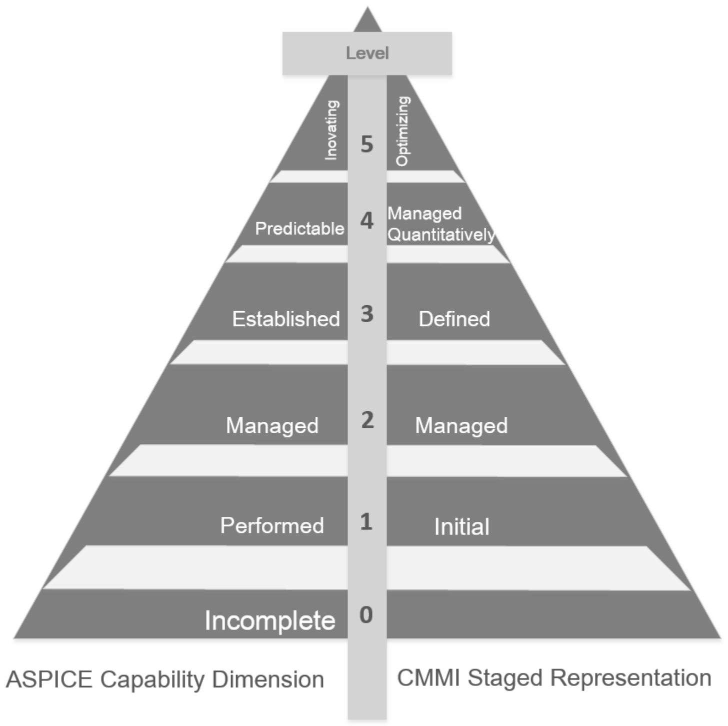


Figure 38 - Representation and comparison on CMMI & ASPICE models

Considering the early stage of the process, establishing high expectations would not be realistic. More than a rate, both CMMI and ASPICE appraisals mainly intend to identify gaps and deficiencies, and provide an improvement path towards the automotive development.

Therefore, the process shall be matched and assessed according to the primary level of each model. Since the Incomplete and Initial levels both represent ad-hoc and chaotic procedures, the target shall comprise Performed and Managed levels for ASPICE and CMMI, respectively.

An important decision on the assessment methodology lies on the measurement. While CMMI does not define a specific rating level for its practices, ASPICE suggests the “NPLF”, already described on Appendix I. In order to achieve compliance between appraisals, this same rating scale shall be used for both models. Considering this overall strategy, the following topics detail the assessment strategy for each model, as the process areas related to the targeted levels.

3.4.1 CMMI

According to the strategy previously described, the assessment conducted over CMMI targets the *Managed* level within its staged representation. Every stage is composed by a predefined set of process areas, which must be performed in order to accomplish the correspondent level. Therefore, in order to achieve the targeted level, the following process areas must be considered:

- REQM- Requirements Management: Management of work product requirements and specification of its components, ensuring the alignment with the plan established for the project.
- PP- Project Planning: Establishment of consistent plans related to project activities and work products, including schedule, resources estimation, and risk identification.
- PMC- Project Monitoring and Control: Monitorization of the progress and status of the project, so corrective actions may be taken when performance deviates significantly from the plan.
- SAM- Supplier Agreement Management: Definition of acquisition processes, involving comparison of appropriate suppliers, and selection of the most suitable products.
- MA- Measurement and Analysis: Development and maintainability of a measurement approach to support management information needs.
- PPQA- Process and Product Quality Assurance: Establishment of an objective insight into processes and associated work products, and its compliance with established standards and procedures.
- CM- Configuration Management: Supervision and control of the integrity of work products, including tools, designs, software, and documentation.

These process areas represent a collection of generic and specific goals, further reflected on generic and specific practises. Specific goals and practises are essential to achieve a determined process area. Generic goals and practises are mutual to several process areas, and describe activities to institutionalize the process within the organization. Both categories shall be considered and matched with the implemented process.

3.4.2 ASPICE

Despite having similar capability levels, ASPICE does not define a set of process areas to be accessed. Such flexibility enables to focus on specific needs of the project, but hinders the comparison to other appraisals. Therefore, in order to follow a common standard within automotive industry, the set of process areas contemplated by the HIS scope has been selected. Accordingly, five process groups shall be addressed:

- ACQ- Acquisition Process Group: Processes performed by the costumer, or by the supplier when acting as a customer for its own suppliers, in order to acquire a product and/or a service.
- SYS- System Engineering Process Group: Processes addressing the elicitation and management of customer and internal requirements, definition of the system architecture, and integration and testing on the system level.
- SWE- Software Engineering Process Group: Processes addressing the management of software requirements and development of the corresponding software architecture, design, implementation, integration, and software testing.
- SUP- Supporting Process Group: Processes that may be employed by any of the other processes at several points along the life cycle.
- MAN- Management Process Group: Processes eventually used by anyone responsible to manage the project or its process.

Each process group comprises one or more processes to be appraised. In order to accomplish the primary level, their process attributes must be largely of fully achieved. Therefore, the assessment shall address each process, and match its attributes with the Agile approach implemented along the project.

4. RESULTS AND DISCUSSION

After specifying the development process, this chapter intends to present the primordial outcomes of its implementation. Moreover, results from comparison with the automotive standards provide concrete information on future improvement strategies.

4.1 PROCESS IMPLEMENTATION

Scrum development process and its supporting practices were successfully employed within the project Bosch Innovcar: "The Cockpit of the Future"[6]. After an initial experiment, the framework has been established with iterations of two weeks. Since the official project kick off, 11 sprints have been performed, each one comprising the Scrum regular activities. In view of that, the diagram in Figure 39 depicts the major stages along the process.

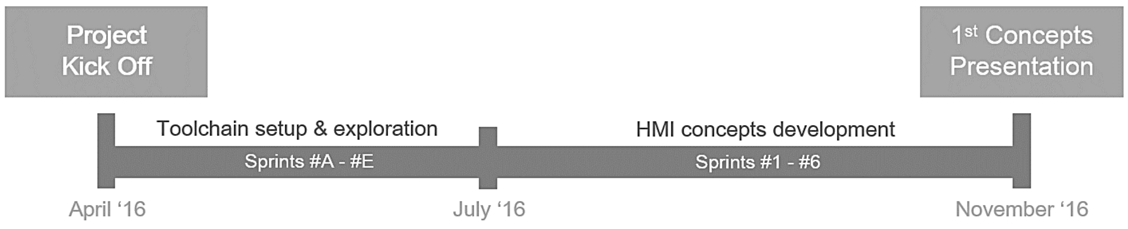


Figure 39 - Bosch Innovcar - Temporal diagram

The initial iterations were focused on the toolchain and its configuration. In order to setup the embedded environment, the operating system and its supporting modules were continuously trialled until achieving a stable version.

[6] A detailed overview of the implemented process and conducted activities is presented on Appendix V.

More than preparing the development framework, this experimental period constituted an opportunity to explore its potentialities and identify eventual limitations. As a result, the experience and knowledge acquired along this phase were determinant to define new concepts and evaluate its feasibility.

In parallel during this initial phase, several cooperative activities with Bosch including workshops, core team meetings, and discussions on the DSM vision enabled the perception of potential concepts. Subsequently, after the 5 early sprints the toolchain was set, and some pilot concepts were ready to be developed.

On a further stage, the novel HMI concepts were introduced iteratively, and reviewed after each development sprint. Due to its proximity with implementation, both design and integration teams were also included on development team sprints (Fig. 40). As a result from this mutual plan, the design team anticipated the needed resources for the iteration, while the integration team projected the concepts to be deployed on DSM. The product owner performed a central role on this process. Being the main interface between Bosch and the development team, the product owner was responsible to evaluate the concepts in terms of maturity and value for the customer. Accordingly, concepts were refined, prioritized, and planned before being introduced on development iterations.

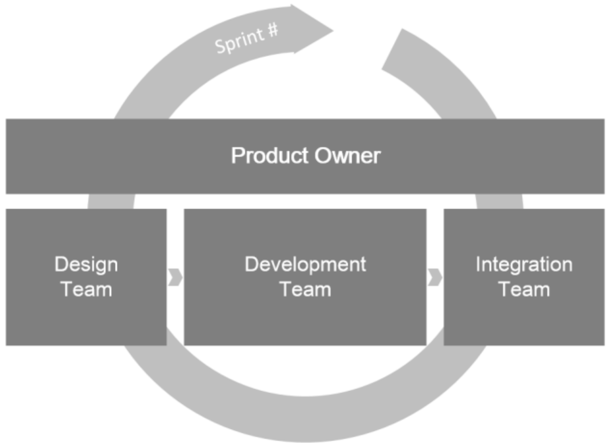


Figure 40 - Iterating teams

Despite the youth of the process and few iterations fully dedicated to development, so far the results are quite optimistic. During the 6 development sprints, 37 stories have been planned, estimated, and duly implemented. Several concepts were addressed, regarding HMI personalisation, warning strategy, infotainment, and the inclusion of personal devices. Since they are closely related to the work packages, these functionalities constitute a solid basis for further iterations.

In order to clarify the process and better understand the product management stages during the iteration, a sprint will be used as an example. Following topics address the events and consequent results of a development sprint #3, occurred from 5-16 September.

4.1.1 SCHEDULING THE ITERATION

Sprints are organized according to the events and procedures delineated by the Scrum guide [40]. As a result, every iteration starts with an initial planning, is continuously inspected through daily scrums, and ends with a global review and a retrospective session. Since sprints were defined with a two-week length, these events were programmed and time-boxed as depicted in the Figure 41, presented bellow.

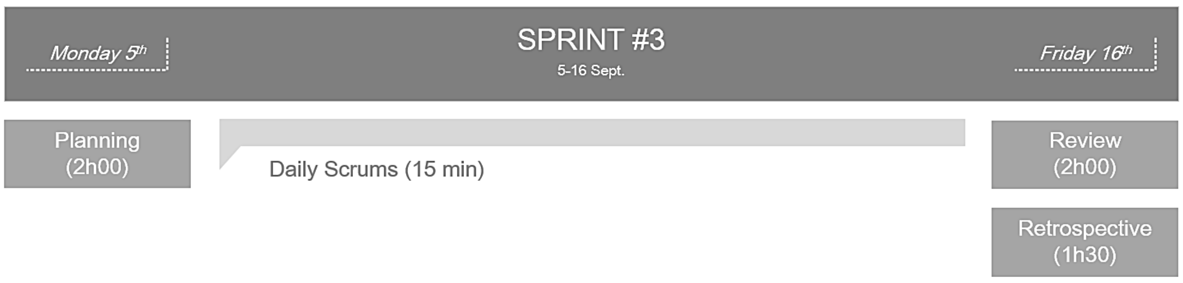


Figure 41 - Sprint #3: Iteration schedule

Although events are performed on specific days, their schedule is determined by the scrum master and agreed by all team members. Nevertheless, the agreed routine must be permanent and rigorously attended by everyone involved.

From the author’s perspective, the product management work during the sprint may be divided in three distinct actions: planning, tracking, and review. Ensuing topics describe each stage, and how it was reflected on sprint #3.

4.1.2 PLANNING THE WORK

Every sprint starts with a planning meeting, which intends to define the work to be performed during the iteration. The event is conducted by the product owner, who describes the product backlog items in detail and defines an overall goal for the sprint. Being responsible for maximizing the product value, the product owner must maintain the backlog prioritized according to customer needs and specifications. The Figure below demonstrates the backlog status early in the sprint #3.

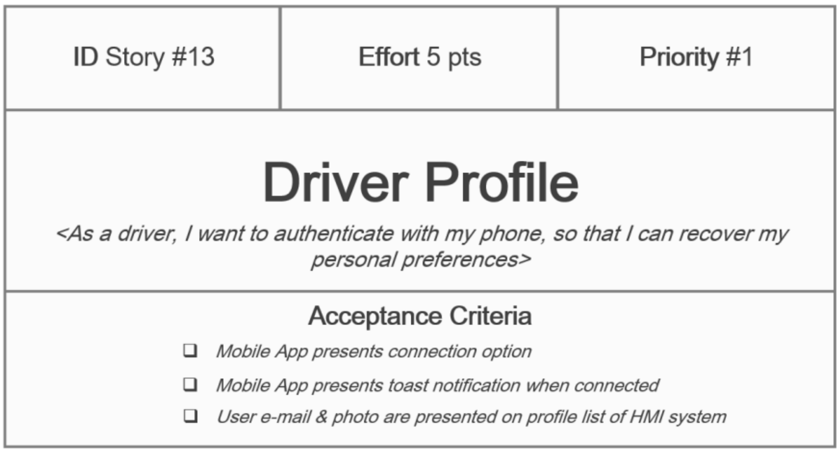
Priority	Story Theme
1	Driver Profile
2	Mobile App: Control Car Functionalities
3	Lower Stack Customization
4	Shortcut Bar
5	Android on Passenger Display
6	Object Between Displays
7	Contacts Synchronization
8	Direct Interface on Music Menu
NP	Fingerprint Authentication
NP	Access Social Networks
NP	Personal Reminders

Figure 42 - Sprint #3: Product Backlog

After an explanation on each item by the product owner, the team discusses internally and defines how many features may be implemented during the iteration. During the sprint #3, the development team committed to deliver the first six stories, which were immediately included on the sprint backlog.

Nevertheless, the items on the product backlog are not detailed nor self-explainable. In order to provide an effective guidance during the iteration, items on the sprint backlog are accompanied by a user story description, priority, estimation points, and acceptance criteria. While estimation points provide an idea on the remaining work during the sprint, the acceptance criteria intends to clarify the user stories and establish boundaries for its development.

As an example of the procedure, following user story card was filled with information correspondent to the first item of the product backlog, named "Driver Profile" (Fig. 43).



ID Story #13	Effort 5 pts	Priority #1

Driver Profile

<As a driver, I want to authenticate with my phone, so that I can recover my personal preferences>

Acceptance Criteria

- ❑ *Mobile App presents connection option*
- ❑ *Mobile App presents toast notification when connected*
- ❑ *User e-mail & photo are presented on profile list of HMI system*

Figure 43 - Sprint #3: Story card

Now with concrete information on the sprint backlog, the team is able to decompose each item into specific tasks of implementation. Although knowledge earned along the sprint may result in unpredicted work, the anticipation of evident tasks promotes structured thinking, and establishes a guidance for development. During the discussion, agreed tasks are written on commonly used post-its and placed on the scrum board, which must be visible and clear to the entire team (Fig. 44). The end of the planning meeting means an agreement on the committed stories, and a detailed plan for their development.

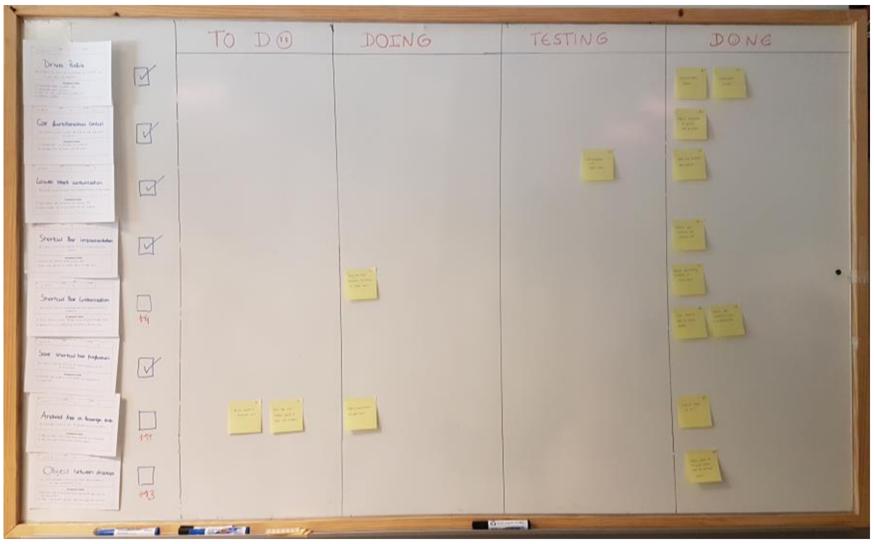


Figure 44 - Sprint #3: Scrum board

4.1.3 Tracking Development

The development is continuously inspected through daily scrums. Usually performed early during the day, these short meetings intend to present recent progressions, report difficulties or eventual impediments, and establish an immediate plan for the current day. As a stand-up event, daily scrums are performed right near the scrum board, so the team may discuss the tasks and update their status. In this sense, the scrum board performs an essential role on tracking development, since it provides an actual standing of development and its progress towards the sprint goal.

Moreover, evaluating the percentage of completed tasks within a story enables the estimation of remaining points, and consequent drawn of burndown charts. Accordingly, the product owner frequently verifies the status of the user stories, so the performance of the team can be measured and registered to further discussion. As an example, the burndown chart resultant from sprint #3 is presented in the Figure 45.

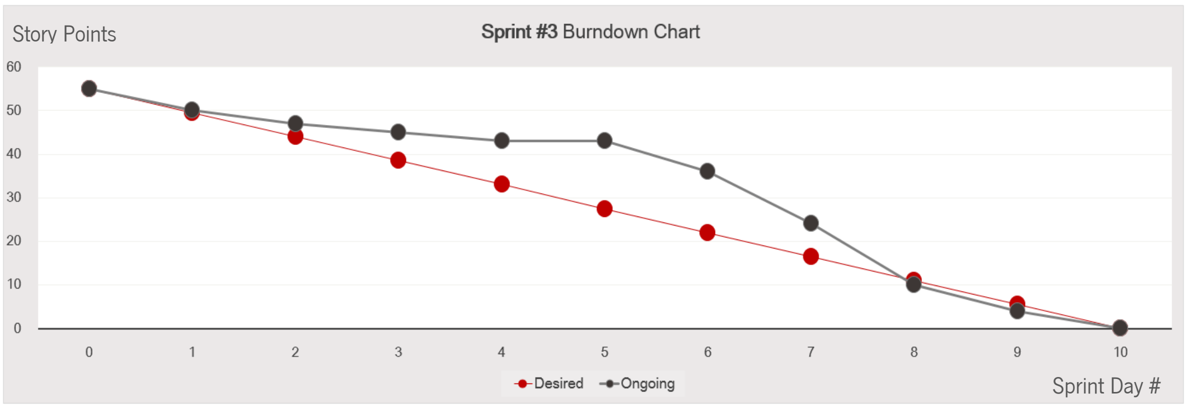
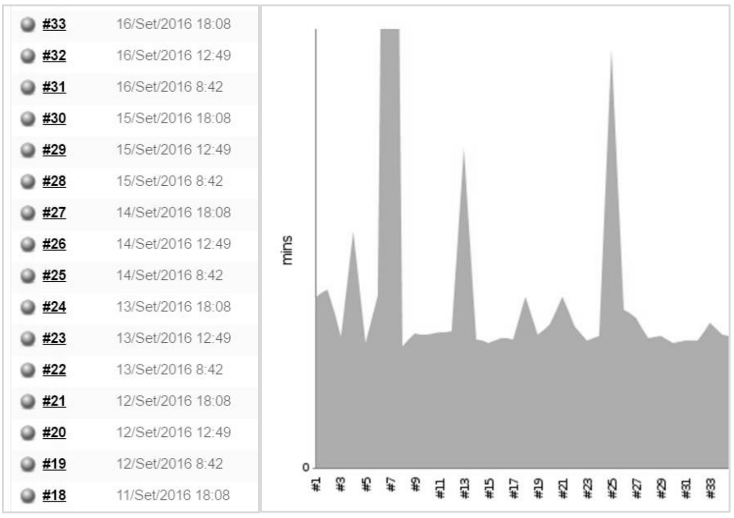


Figure 45 - Sprint #3: Burndown chart

Considering the accomplishment of the committed work, the Sprint #3 successfully achieved its goals. The stabilization at the middle of the sprint suggests lack of progress, due to unpredicted bugs and consequent additional tasks. In fact, it shows the debilities of work estimation since there are several variables that cannot be anticipated, such as bugs, lack of knowledge, and technological deficiencies. As a result, the effort spent on solving these issues is negatively reflected on sprint results. Nevertheless, burndown charts provide relevant information on team capabilities, and may be used as learning basis for future estimations.

Besides the performance and task execution, development is also tracked on the software level. The implemented setup of continuous integration tools provide data on versioning, building history, compilation times, etc. Allocated on a local server, Jenkins presents a complete dashboard with relevant information, which can be consulted anytime during the iteration. Accordingly, a new workspace is created for every sprint, so the committed changes and building registries can be easily consulted. Following chart in Figure 46 shows the Jenkins partial outputs relative to sprint #3.



#33	16/Set/2016 18:08
#32	16/Set/2016 12:49
#31	16/Set/2016 8:42
#30	15/Set/2016 18:08
#29	15/Set/2016 12:49
#28	15/Set/2016 8:42
#27	14/Set/2016 18:08
#26	14/Set/2016 12:49
#25	14/Set/2016 8:42
#24	13/Set/2016 18:08
#23	13/Set/2016 12:49
#22	13/Set/2016 8:42
#21	12/Set/2016 18:08
#20	12/Set/2016 12:49
#19	12/Set/2016 8:42
#18	11/Set/2016 18:08

Figure 46 - Sprint #3: Jenkins build history & comparison of compilation times

Both automated and version triggered builds are registered by the integration tool, which provides a detailed console output for every compilation event and a visual indicator on software stability. Therefore, due to its relevance and support on continuous integration mindset, Jenkins performs an essential role within the development.

Being responsible for the product, the product owner must be aware of its condition during the sprint. Leading the team to the desired results, however, cannot be automated. It requires accompaniment, discussion, and continuous contact with the team. By closely collaborate on an ongoing basis, the product owner acquires a clearer view of their concerns, needs, and strategies; so immediate help and guidance may be provided.

Gathering data during development is only useful when used to improve the product, organization, or its procedures. Accordingly, sprint measures and results are specifically addressed at the end of the iteration, as subsequently described.

4.1.4 Review and Retrospectives

Finalizing two weeks of development, the iteration closes with an inspection event. Results and outcomes of the completed sprint are presented on a review meeting, which may include customers, stakeholders, and representatives from other teams. Aware of the accomplished goals along the iteration, the product owner may request specific demos and prototypes to demonstrate the essential features. Without focusing the technical approach, the product and its functionalities may be conferred. Besides eliciting feedback on past work, such discussion promotes a critical appreciation that may serve as a starting point to plan the next sprint.

After discussing and inspecting the product, the retrospective meeting provides an opportunity for the team to inspect itself. As an internal event, the retrospective addresses the flow of the previous sprint, regarding relationships, procedures, and tools. Problems and any occurred difficulties shall be mentioned, as they constitute possible areas of improvement. When invited, the product owner may discuss on achievement of the iteration goals. In case of missing the committed objectives, the failed estimation shall be evaluated. Moreover, the effort pace measured through the burndown chart also may be addressed. As a diary record, the daily effort chart provides a representation on the pace of the team. As an example, Figure 47 shows the resultant chart of sprint #3.

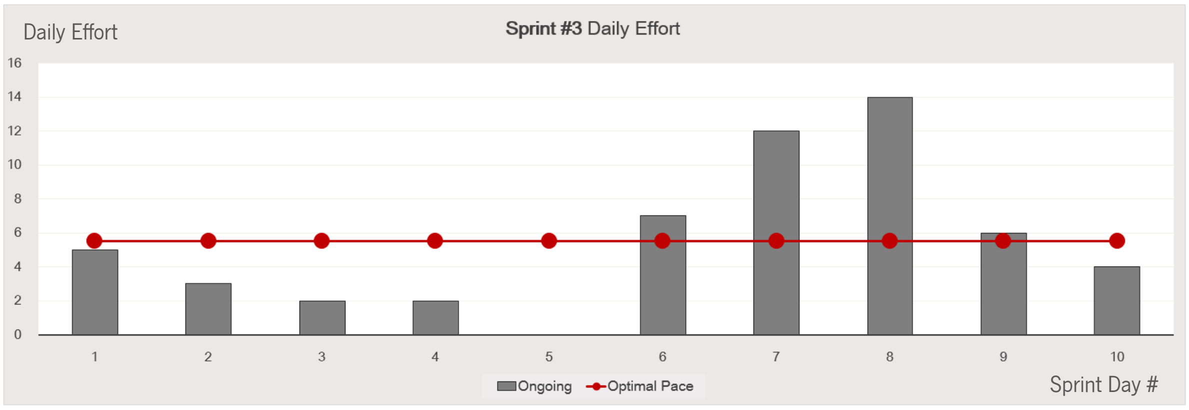


Figure 47 - Sprint #3: Daily effort chart

As previously explained, the effort spent on day #5 was dedicated to bug solving. Nevertheless, the effort clearly increased on the second half of the sprint, suggesting that the work organization and division along the sprint may be improved. In fact, this example reveals the purpose of retrospectives: improvement. Through open and transparent discussions on flaws and difficulties, the team is able to identify potential enhancements, and proceed to its employment on the next iteration.

4.2 Automotive Compliance

Standards as CMMI and ASPICE have become imperative nowadays, since they constitute the basis to achieve consistency and consequent recognition within the automotive industry. Despite being on an embryonic stage, the implemented development process was matched and assessed for both models. More than simply achieving a determined level, the appraisals intended to detect process gaps and major debilities. As a result, an improvement plan may be furtherly defined according to both organization and project needs.

From a global perspective, Agile and Scrum provide an acceptable response to CMMI and ASPICE requirements, especially on management and control domains. On the other hand, there are several process areas which are uncovered and may require additional practices, mainly regarding quality and monitoring fields. Although some results are identifiable on both models, CMMI and SPICE follow a different structure and address particular areas. Therefore, the ensuing topics present the outcomes and consequent analysis for each model.

4.2.1 CMMI

Being the most recognizable framework for improvement, CMMI comprises a certain degree of flexibility that makes the model applicable to any process or organization. Rather than defining a procedure, CMMI states the core activities to be performed. Thus, even an early process may benefit from its appraisals, and use them as improvement guidelines.

The established goal was to achieve the “Managed” maturity stage, which involves control of expected outputs, collaboration with stakeholders, management, and constant review of the process. In order to accomplish this second stage, CMMI defines seven process areas to be achieved: Requirements Management (REQM), Project Planning (PP), Project Monitoring and Control (PMC), Supplier Agreement Management (SAM), Measurement Analysis (MA), Process and Product Quality Assurance (PPQA), and Configuration Management (CM). Each process area comprises a predefined set of specific goals and correspondent practices, which must be satisfied in order to achieve the desired maturity level.

Accordingly, the process areas previously described were evaluated and matched with the implemented process. Detailed results comprising their description and consequential achievement are provided on Appendix VI. Nevertheless, Figure 48 presents an overall view of the assessment outcomes.

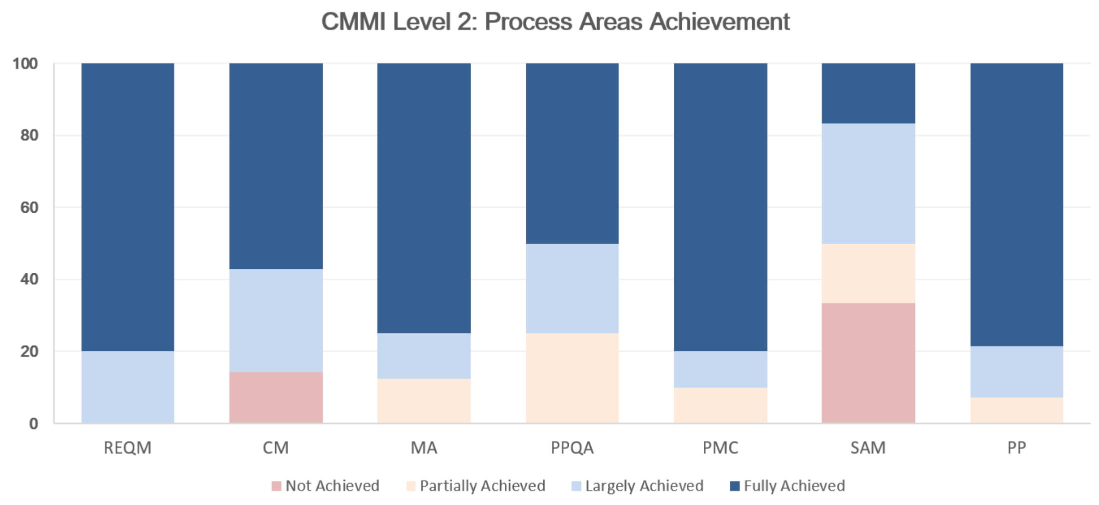


Figure 48 - CMMI appraisal results

Amongst the evaluated process areas, some positive results were distinguished. Several areas were generally satisfied as Requirements Management, Project Planning, and Project Monitoring and Control. Despite having different practises, they are mostly related to management activities such as requirements gathering, work planning and monitorization, and development review. As previously discussed, these activities are intrinsically present on the Scrum framework. Practices and events as planning, daily scrums, grooming sessions, and reviews, ensure an iterative planning and continuous monitorization of the process. Consequently, the results under the correspondent process areas achieved a positive evaluation.

In other hand, some process areas are not directly covered by the Scrum methodology, and as a result were negatively reflected on the assessment. Supplier Agreement Management, regarding acquisition procedures and supplier selection, represents the process area with more unachieved practices. It is followed by Configuration Management, which embraces supervision of the work products and documentation. They constitute an evident gap on the process, mainly due to its absence on Agile methods and its focus on development. Supplier Agreement Management process area is not crucial for the process, since the project scope is narrowed to investigation and innovative development, so buying processes are mainly managed by the costumer. However, Configuration Management definitely constitutes an area to progress, due to the importance of supporting tools and documentation on the development process.

Finally, Process and Product Quality Assurance process area also presents a long way to improve. This area implicates an objective evaluation on both process and products compliance, and consequent resolution for nonconformity issues. Besides continuous inspection, explicit quality reviews are not covered by the Scrum development process, and constitute another field to improve.

Therefore, while Scrum and supporting practices covered the majority of the goals established by CMMI, some process areas and their related practices require additional attention. Nevertheless, the development process presents a positive rate, as presented in the overall chart of Figure 49.

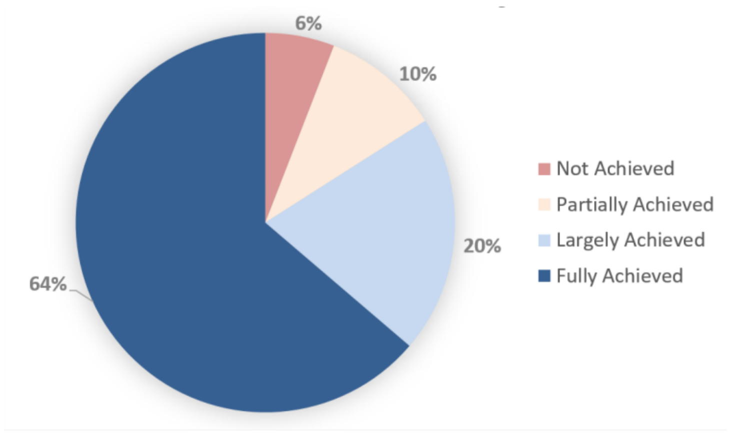


Figure 49 - CMMI level 2 global achievement

The results are optimistic, as nearly 75% of the process areas are largely of fully achieved. However, more than creating the expectations on achieving a maturity level, these results show that there is still a long way to accomplish it. Although only a few process areas require full implementation, the entire group can be significantly enhanced and conduced towards a more effective process.

4.2.2 ASPICE

Contrasting the previous model, ASPICE is exclusively centered on the automotive industry. Thus, rather than defining overall practices applicable to every domain, ASPICE processes are centred on automotive specific needs. Accordingly, considering the evolution and actual trends on this demanding industry, ASPICE puts a special focus on software and embedded systems development. Such emphasis on automotive development definitely suited the scope of this project, and represented an effective approach to identify crucial areas of improvement.

Since no predefined processes are imposed by the ASPICE reference model, the organization must define which areas shall appraise. In order to follow a common approach within the automotive industry, this assessment selected the recognizable set of process areas contemplated by the HIS scope. Besides being mainly focused on software engineering processes, the HIS scope also includes acquisition, supporting, and management domains. Each of these groups contain several processes, composed by sets of base practices and resultant work products.

The established goal was to achieve the first capability dimension, entitled the "Performed Process". Despite being the primordial level in the progressive scale, the performed dimension requires an accurate implementation of the process. In order to achieve it, the entire group of processes and related base practices must be largely of fully achieved. Accordingly, its content was evaluated and matched with the implemented process. Detailed results comprising their description and consequential achievement are provided on Appendix I. Nevertheless, following charts present an overall view of the assessment outcomes for each process group.

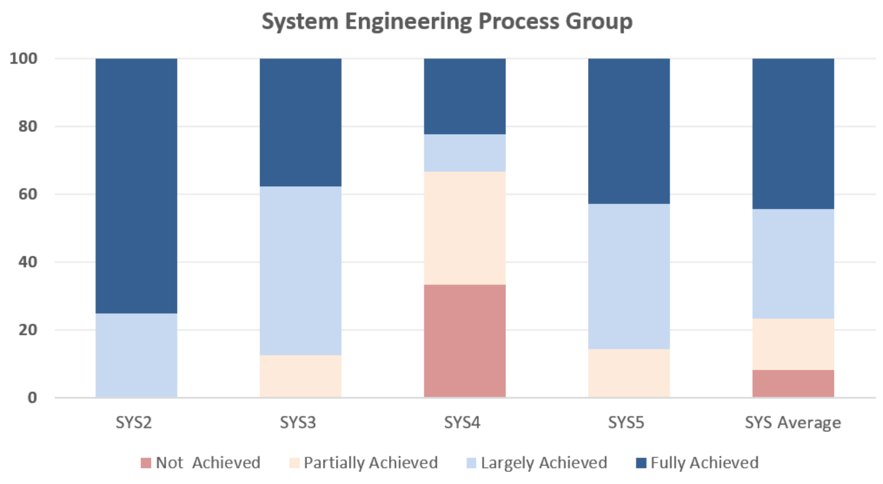


Figure 50 - ASPICE SYS appraisal results

The system engineering process group (Fig. 50) addresses the definition of the overall system, involving both customer and internal requirements (SYS2), architectural design (SYS3), integration (SYS4), and qualification tests (SYS5). While requirements elicitation and analysis is significantly covered by Scrum, the development process lacks on specific procedures for the remaining areas, mainly regarding integration processes. Despite having a positive average on its process areas, this system engineering process group definitely needs further attention. Moreover, the appointed debilities also become evident on the subsequent process group, software engineering, whose results are presented next.

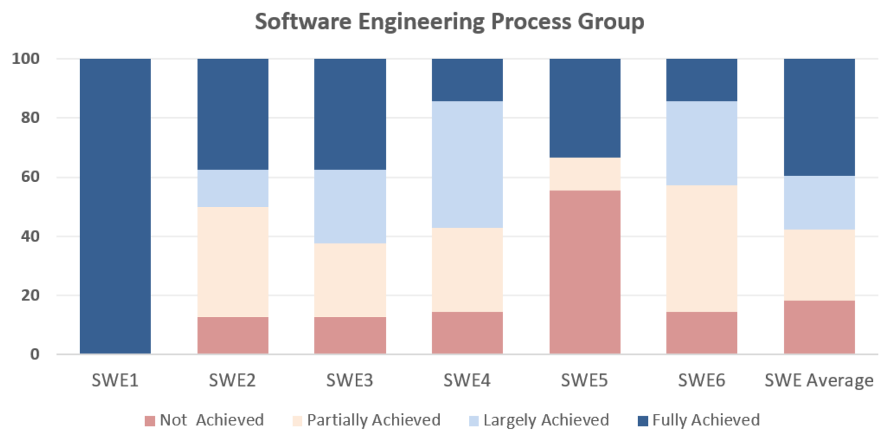


Figure 51 - ASPICE SWE appraisal results

The software engineering process group (Fig. 51) addresses the entire course of development, regarding requirements analysis (SWE1), architectural design (SWE2), design (SWE3), unit verification (SWE4), integration (SWE5), and qualification test (SWE6). Although their average shows a considerable achievement, nearly 60% of the base practices still need improvement.

Software requirement analysis is clearly the highest rated, mainly due to specific procedures delineated by Agile that target its specification. Events such as planning, grooming, and user story gathering provide a consistent framework so requirements can be collected and analysed iteratively. In the other hand, practices explicitly related to software development are not specified by Scrum, such as integration procedures, architectural designs, tests, and quality measurements. Some base practices regarding these areas were achieved through acceptance testing and integration tools which were additionally implemented. However, in order to largely accomplish the entire software engineering process group, some consistent procedures concerning architectural design, unit testing, and integration must be employed.

Transitioning into a broad context, the supporting life cycle category comprises processes that may be employed by the previous engineering areas. Therefore, rather than specifying additional procedures, these processes intend to support both system and software development, addressing aspects as quality assurance (SUP1), configuration management (SUP8), problem resolution management (SUP9), and request management (SUP10). The results regarding this initial assessment are presented in the following chart.

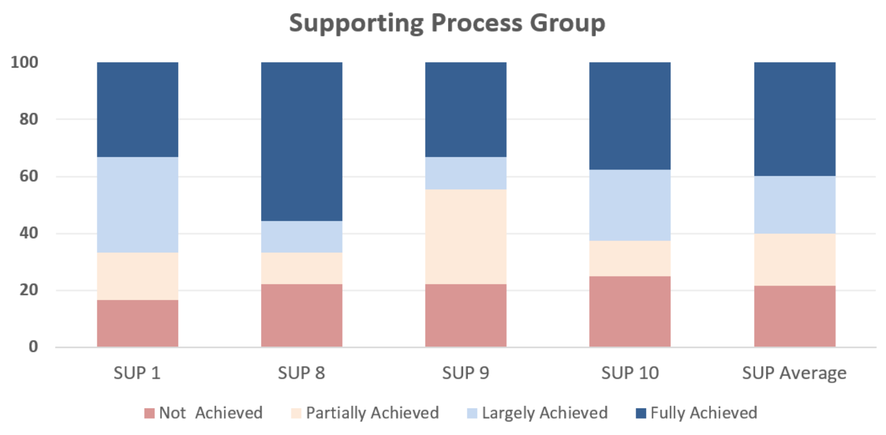


Figure 52- ASPICE SUP appraisal results

Considerable debilities on supporting category (Fig. 52) are shown through this results. Problem resolution management (SUP9) reveals the lowest percentage of fully achieved practices, mainly due to lack of tracking and analysis procedures. In fact, this deficiency constitutes the principal weakness of the entire group. The majority of the base practices is considered within the development process, as problem identification, control of work products, and continuous discussion. However, these procedures are not repeatable and reliable to be fully achieved. Moreover, in order to achieve the desired consistency, these supporting processes require documentation and effective reports on every action. Although avoiding excessive documentation is one of the main Agile principles, this is a necessity to be furtherly addressed. In fact, such decisions are directly related with the following categories, Management and Acquisition, whose results are presented in the Figure 53.

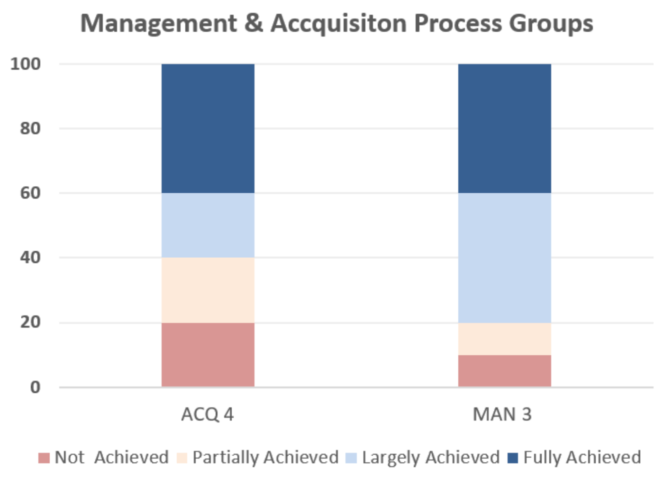


Figure 53- ASPICE MAN & ACQ appraisal results

Acquisition and management are distinct categories, each one represented by a single process. The acquisition process group addresses relations and procedures between customer and supplier. The supplier monitoring process (ACQ4) particularly involves tracking and assessing the performance of the supplier against agreed requirements. Thus, the procedure is mainly applicable when the acquisition comprises software units, or specific designed components. Since this development process is applied to an investigation project, and the majority of its acquisitions are COTS, these procedures may not be essential.

Finally, the management category consists of processes regarding the project roadmaps and goals on an organizational level. The project management process (MAN3) involves to identification, establishment, and control of the activities and needed resources, so the project creates the desired product. Management activities involving the project life cycle, feasibility, resources and knowledge estimation, and progress reporting are considered by this process area. Despite lacking a rigorous schedule of the project, Scrum framework comprises planning, estimation, and reporting actions. Therefore, rather than an exhaustive intervention at the beginning of the project, the project is iteratively monitored with collaboration and control of the management. Considering the entire set of categories and correspondent processes, the development process obtained the overall achievement presented in the following chart (Fig. 54).

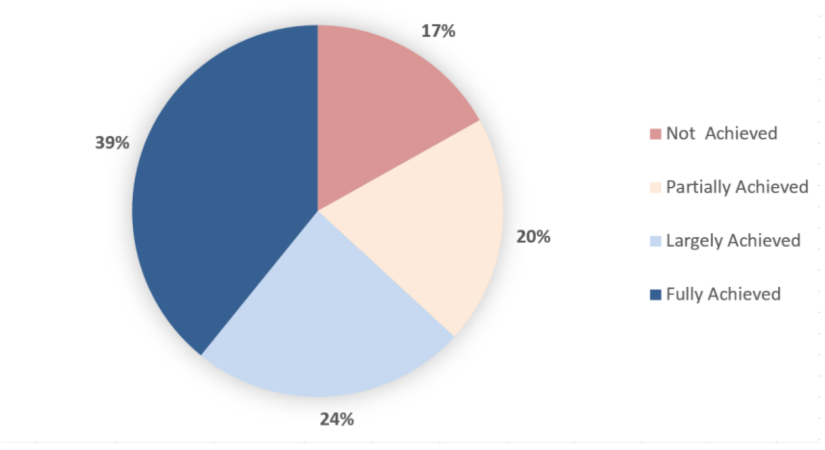


Figure 54 - ASPICE level 1 global achievement

Overall results indicate a considerable adherence to ASPICE processes, since nearly 65% of their subsequent practices are fully or partially achieved. In other hand, 17% of the areas show no evidence of implementation, while 20% still are unpredictable and just partially achieved. Therefore, several processes and related practices must be improved in order to accomplish the desired level.

5. FINAL CONCLUSIONS

Technological evolution brought serious challenges to automotive industry. In order to keep pace with innovation and react to market demands, automotive is under constant pressure to modernise and embrace new technologies. As a result, the same vehicles that once were pure mechanical progressively turned into sophisticated systems, mainly built over software and embedded units. Their dependencies and increasing complexity constitute a prominent obstacle for the automotive industry, evinced by the both long and expensive process of launching a new product. Despite being partially explained by its rigorous standards, these challenges show an urgent need to enhance automotive development processes.

Analysing other domains where software performs an important role, a solution was hypothesized through the use of Agile processes. Employing an action research methodology, the author addressed an immediate practical problem while developing scientific knowledge on this subject. Accordingly, Agile was evaluated as a possible solution to automotive challenges through an active participation on an automotive HMI development project. After an initial phase of research and study on Agile methods, a development process was designed, and applied to an actual investigation project. Finally, the implemented process was compared and evaluated according to models highly recognized by the automotive industry, namely CMMI and ASPICE. While data gathered along the development enabled to draw concrete conclusions on benefits and drawbacks of Agile, matching the process with the standards clarified its suitability for automotive industry.

5.1.1 LEARNINGS

The literature revised along this work clearly demonstrated the progression of Agile. Built over iterative and flexible values, the framework is an absolute advantage for every domain where requirements and market trends are volatile. Accordingly, several methods have been purposed, and their benefits are nowadays being attested by numerous industries. Such diversity evinces a fundamental conclusion of this work: rather than a controlling methodology, Agile virtue remain on its principles. The process and its practices may consider the needs of a specific field as automotive, but success entirely depends on changing the mindset to Agile values. In fact, that was the leading conclusion of the action taken on this work. Centered on development of automotive HMI, the project was early divided into work packages and prioritized according to value for innovation purposes. Although the ordering involved other international departments of Bosch with a broader view of the technologies and the current market, the priorities were realigned a few months after its first establishment. Nevertheless, since requirements are gathered on an iterative approach, Agile encourages communication and openness to redefine the plan. Such flexibility places Agile closer to business goals, and represented a clear advantage on this investigation project.

Subsequently, Scrum was the elected methodology to implement. Despite prescribing specific roles and procedures, the objectivity of the framework made Scrum a balanced and suitable solution for this project. Consequently, development was organized in sprints of two weeks, which enabled an effective tracking without spending excessive time on meetings. The team was organized according to Scrum roles, and specific events as sprint planning, daily scrum, sprint review, and retrospective were rigorously performed. Besides providing opportunities for discussing and inspecting the work, these events promoted regular communication and enhanced the development process. Although Scrum rigorously defines organizational and development tracking procedures, the framework lacks specific practices for development. Therefore, additional actions regarding validation and continuous integration were added to the process. Suitable tools were selected according to both team and project necessities, always taking into account the resultant overhead to the process. While acceptance testing clarified the requirements for implementation, version control and automated tools made development more consistent.

Besides defining the process, the author took an active part on managing the development. As a product owner, the author discussed the product roadmap, gathered requirements, and acted as an interface between the customer and the development team. Features were prioritized, estimated, and tracked along each iteration. Therefore, the participation on Scrum events constituted a fundamental part of the work. More than guiding the development, Scrum ceremonies were regular opportunities to detect process weaknesses, which were immediately resolved or appointed for further discussion. Thus, being part of the process provided a broader view of Agile methods, and clarified the major benefits and drawbacks of implementing a framework as Scrum.

Moreover, the active role on the process also represented an advantage when evaluating its compliance with automotive standards. The implemented process was appraised and compared with two models of great importance within the automotive industry, namely CMMI and ASPICE. The assessment was divided in process areas, whose general achievement is measured in maturity or capability levels. Nevertheless, more than achieving a determined level, the assessment intended to provide an insight on the implemented process and its shortcomings towards the automotive industry. Accordingly, the results of both models exposed a considerable number of partially achieved or even not achieved practices. Deficiencies mainly regarding testing and integration procedures were mutually identified, and constitute crucial areas to improve. In addition, several procedures concerning design and quality assurance are fairly addressed, but documentation and progressive recordings are lacking on the process.

Although both models generally identify the correspondent deficiencies of the process, ASPICE results are considerably inferior. Nevertheless, the difference only reflects the dissimilarity of the models and their assessment strategy. While CMMI defines a group of processes for each maturity level, ASPICE does not prescribe any processes to appraisal. Therefore, this work considered the entire group of processes comprised by the HIS scope, which covers a wider range than the process areas of CMMI. Moreover, since ASPICE is explicitly targeted for automotive development, its processes areas are far more specific than a general model as CMMI. Thus, despite being more negative, ASPICE results are beneficial in the sense they provide a clearer idea of the process capabilities and deficiencies within the automotive standards.

Consequently, the conducted assessments show a long way to improve towards an effective development. The implemented process clearly suits the investigative purposes of the project, as it enables constant feedback and rapid prototyping. Sprints enable interaction and collaboration of the several teams on creating innovative concepts. Accordingly, Agile framework has evinced great advantages on iterative requirements, costumer involvement, and continuous delivery. However, in order to eventually meet the automotive standards for a development process, the methodology must be strengthened with additional practices.

5.1.2 Future Work

Further steps of this work may be divided according to both domains of action research methodology: the practical case study, and the scientific knowledge. Focusing on the Innovcar project, there are several procedures within the development process to improve. Due to the considerable number of teams involved, the interface between the research groups and the multidisciplinary team remains undefined. The design team constitutes the bottleneck for development, so these interactions must be further delineated.

Scrum framework was duly implemented within the development team; however several procedures still can be enhanced. As an example, the subjectivity of story estimation complicates the development tracking. Yet, such effort may be rewarded since the accuracy of the estimates also tend to improve with team experience. Additionally, supporting tools shall be implemented for automated testing, requirements gathering, and development tracking. Besides providing statistical data on development, these management tools typically support documentation and testing procedures, which constitute an evident deficiency of the process. Nevertheless, any change on the process must ponder the Agile main principles, which prevent the overhead for excessive tools and documentation. Rather than fixing the process, the focus shall always be to continuously improve its agility.

Concluding with the broad perspective of the scientific and technological scope of this work, Agile has strong arguments to become a solid approach within the software industry. Flexibility and customer involvement are desirable qualities of a framework, which is capable to maintain the teams motivated while receiving encouraging feedback. Nevertheless, this characteristic flexibility is the exact opposite of a demanding industry such as automotive. High safety and quality standards are not contemplated for a raw framework as Scrum. Therefore, an interesting area of research would be to combine the best of both worlds: quality and validation procedures from automotive, with iterability and involvement from Agile. The result would be a robust and consistent Agile based framework, plainly suitable for a demanding industry as automotive.

Bibliography

[1] M. Hoelz, M. Collings, and H. Roehm, "A new era Accelerating toward 2020 — An automotive industry transformed," pp. 1–32, 2009.

[2] A. Davydov, "Automotive HMI Fit for 2020," *Technol. Excell. Ser. Key trends Affect. Evol. in-car user interfaces*, 2012.

[3] NXP AMPG Body Electronics Systems Engineering Team, "Future Advances in Body Electronics AMPG Body Electronics Systems," 2013.

[4] D. Durisic, M. Nilsson, M. Staron, and J. Hansson, "Measuring the impact of changes to the complexity and coupling properties of automotive software systems," *J. Syst. Softw.*, vol. 86, no. 5, pp. 1275–1293, 2013.

[5] A. Busnelli, "Car Software: 100M Lines of Code and Counting." [Online]. Available: https://www.linkedin.com/pulse/20140626152045-3625632-car-software-100m-lines-of-code-and-counting. [Accessed: 06-Apr-2016].

[6] M. Bro, "Challenges in automotive software engineering," vol. 2006, pp. 33–42, 2006.

[7] A. Shaout and G. Waza, "Solutions to Automotive Software Engineering Challenges," *Int. J. Comput. Organ. Trends*, vol. 16, no. 1, pp. 12–19, 2015.

[8] T. (Translogic) Shea, "Why Does It Cost So Much For Automakers To Develop New Models?" [Online]. Available: http://www.autoblog.com/2010/07/27/why-does-it-cost-so-much-for-automakers-to-develop-new-models/. [Accessed: 21-Feb-2016].

[9] Hp and V. Uk, "Case Study: Vodafone UK and HP partnership more than halves the software development lifecycle."

[10] HP, "Agile is the new normal," pp. 1–3, 2015.

[11] B. J. Ehlert and S. Manager, "Agile Quality Automation Speeds Delivery and Reduces Risk."

[12] R. L. Baskerville, "Investigating information systems with action research," *Commun. AIS*, vol. 2, no. 3, p. 4, 1999.

[13] T. Cornford and S. Smithson, *Project Research in Information Systems*. 1996.

[14] N. F. Kock, *Information systems action research : an applied view of emerging concepts and methods / edited by Ned Kock*. 2007.

[15] R. N. Charette, "This Car Runs on Code," *IEEE Spectrum*, 2009. [Online]. Available:

http://spectrum.ieee.org/transportation/systems/this-car-runs-on-code. [Accessed: 06-Dec-2015].

[16] R. W. Cox, "GM Emission Control Project Center - I Was There - Generations of GM." [Online]. Available: https://history.gmheritagecenter.com/wiki/index.php/GM_Emission_Control_Project_Center_-_I_Was_There. [Accessed: 06-Dec-2015].

[17] D. W. W. Royce, "Managing the Development of large Software Systems," *Ieee Wescon*, no. August, pp. 1–9, 1970.

[18] P. Rook, "Controlling software projects," *Softw. Eng. J.*, vol. 1, p. 7, 1986.

[19] The Standish Group, "CHAOS MANIFESTO 2013: Think Big, Act Small," *Standish Gr. Int.*, pp. 1–52, 2013.

[20] K. M. C. GmbH, "Agile in automotive - state of practice 2014," no. January, 2014.

[21] "Principles behind the Agile Manifesto." [Online]. Available: http://agilemanifesto.org/principles.html. [Accessed: 12-May-2016].

[22] R. Takahira, L. Laraia, and F. Dias, "Scrum and Embedded Software development for the automotive industry," *... Eng. ...*, pp. 2664–2672, 2014.

[23] "VersionOne: Pioneers in Agile." [Online]. Available: https://www.versionone.com/about/. [Accessed: 10-Dec-2015].

[24] J. Renaudin, "Samsung Moving from Waterfall to Agile to Shorten Galaxy Development." [Online]. Available: https://www.techwell.com/techwell-insights/2015/08/samsung-moving-waterfall-agile-shorten-galaxy-development. [Accessed: 10-Dec-2015].

[25] B. Schatz, I. Abdelshafi, and P. Systems, "Primavera Gets Agile: A Successful Transition to Agile Development," *Ieee Softw.*, vol. 22, no. 3, 2005.

[26] P. Correia and R. Cunha, "CMMI & Scrum at Primavera - A powerfull combination," 2016.

[27] J. Highsmith, *Agile Software Development Ecosystems*. 2002.

[28] J. Highsmith, A. Cockburn, and C. Consortium, "Agile Software Development: the business of innovation," *Computer (Long. Beach. Calif).*, vol. 34, no. 9, pp. 120–127, 2001.

[29] "Manifesto for Agile Software Development," 2001. [Online]. Available: http://www.agilemanifesto.org/. [Accessed: 05-Nov-2015].

[30] N. B. Moe, T. Dingsøyr, and T. Dybå, "Understanding self-organizing teams in agile software development," *Proc. Aust. Softw. Eng. Conf. ASWEC*, no. 3, pp. 76–85, 2008.

[31] A. Cockburn and J. Highsmith, "Agile Software Development:The People Factor," *Computer*

(Long. Beach. Calif)., vol. 34, no. 11, pp. 131–133, 2001.

[32] J. Shore, "The Crucible of Great Teams," *The Art of Agile*, 2008. [Online]. Available: http://www.jamesshore.com/Blog/The-Crucible-of-Great-Teams.html.

[33] J. Shore and S. Warden, *The Art of Agile Development*. O'Reilly.

[34] L. A. Williams and R. R. Kessler, "All I really need to know about pair programming I learned in kindergarten," *Commun. ACM*, vol. 43, no. 5, pp. 108–114, 2000.

[35] L. Williams, R. R. Kessler, W. Cunningham, and R. Jeffries, "Strengthening the Case for Pair-Programming," *IEEE Softw.*, vol. July-Augus, no. August, pp. 19–25, 2000.

[36] A. Cockburn and L. Williams, "The costs and benefits of pair programming," *Extrem. Program. examined*, pp. 223–243, 2001.

[37] S. Stolberg, "Enabling agile testing through continuous integration," *Proc. - 2009 Agil. Conf. Agil. 2009*, pp. 369–374, 2009.

[38] K. Beck, *Extreme Programming Explained - Embrace change*. 2004.

[39] H. K. Flora and S. V. Chande, "A Systematic Study on Agile Software Development Methodologies and Practices," *Int. J. Comput. Sci. Inf. Technol.*, vol. 5, no. 3, pp. 3626–3637, 2014.

[40] K. Schwaber and J. Sutherland, "The scrum guide," *Scrum. org*, no. July, p. 17, 2014.

[41] B. V. De Carvalhoa and C. H. P. Mellob, "Scrum agile product development method-literature review, analysis and classification," *Prod. Manag. Dev.*, vol. 9, no. 1, pp. 39–49, 2011.

[42] T. Ohno, *Toyota Production System*, vol. 4. 1988.

[43] M. Poppendieck and M. A. Cusumano, "Lean software development: A tutorial," *IEEE Softw.*, vol. 29, no. 5, pp. 26–32, 2012.

[44] R. Charette, "Challenging the Fundamental Notions of Software Development," 2002.

[45] D. J. Anderson, *Kanban: Successful Evolutionary Change for Your Technology Business*. Blue Hole Press, 2010.

[46] H. Kniberg, *Kanban and Scrum - Making the most of both*. 2009.

[47] M. O. Ahmad, J. Markkula, and M. Ovio, "Kanban in Software Development: A Systematic Literature Review," *Softw. Eng. Adv. Appl. (SEAA), 2013 39th EUROMICRO Conf.*, no. September 2013, pp. 9–16, 2013.

[48] J. A. Highsmith, *Adaptive Software Development: A Collaborative Approach to Managing Complex Systems*, vol. 12. 2000.

[49] V. Günal, "Agile Software Development Approaches and Their History," 2012.

[50] P. Coad, E. Lefebvre, J. De Luca, and J. de Luca, "Java Modeling In Color With UML," in *Java Modeling In Color With UML: Enterprise Components and Process*, no. c, 1999, pp. 1–12.

[51] DSDM Consortium, "The DSDM Agile Project Framework." [Online]. Available: https://www.dsdm.org.

[52] A. Cockburn, "Crystal Clear. A Human-Powered Methodology For Small Teams, including The Seven Properties of Effective Software Projects," *Integr. Vlsi J.*, p. 39, 2004.

[53] VersionOne, "10th Anual State of Agile Report 2015," 2015.

[54] S. Thomas, "Rugby is a Better Analogy for Agile Delivery than the Scrum | It's a Delivery Thing," 2012. [Online]. Available: http://itsadeliverything.com/rugby-is-a-better-analogy-for-agile-delivery-than-scrum.

[55] Scrum Inc., "The Basics of Scrum: An introduction to the framework."

[56] M. Cohn, "User Stories and User Story Examples." [Online]. Available: https://www.mountaingoatsoftware.com/agile/user-stories.

[57] M. Cohn, "Scrum Product Backlog and Agile Product Backlog Prioritization." [Online]. Available: https://www.mountaingoatsoftware.com/agile/scrum/product-backlog.

[58] "3 Powerful Estimation Techniques for Agile Teams," 2014. [Online]. Available: https://www.sitepoint.com/3-powerful-estimation-techniques-for-agile-teams/.

[59] LeSS - Large Scale Scrum, "Potentially Shippable Product Increment." [Online]. Available: https://less.works/less/framework/potentially-shippable-product-increment.html.

[60] D. Panchal, "What is Definition of Done (DoD)?" [Online]. Available: https://www.scrumalliance.org/community/articles/2008/september/what-is-definition-of-done-(dod).

[61] K. Horvath, "Compliance in Automotive Development," *Intland Software*, 2015. [Online]. Available: https://intland.com/blog/automotive/compliance-in-automotive-development-iso-26262-iec-61508-aspice-cmmi-and-more/.

[62] CMMI, "CMMI for Development, Version 1.3," *Carnegie Mellon Univ.*, no. November, p. 482, 2010.

[63] SEI-Software Engineering Institute, "Capability Maturity Model Integration - Community." [Online]. Available: http://www.sei.cmu.edu/cmmi/.

[64] CMMI Institute, "Background on CMMI," 2016. [Online]. Available: http://cmmiinstitute.com/about-cmmi-institute.

[65] U. Carnegie Mellon, "CMMI ® Executive Overview," *Defense*, pp. 1–41, 2006.

[66] P. Cmm, P. A. Results, T. Standard, C. Appraisal, P. Improvement, P. Cmm, S. Method, D. Document, T. Scampi, and T. P. Cmm, “Introduction to CMMI Appraisals.”

[67] M. Tarnowski, “SCAMPI — Standard CMMI Appraisal Method for Process Improvement,” 2014. [Online]. Available: http://www.plays-in-business.com/scampi-standard-cmmi-appraisal-method-for-process-improvement/.

[68] VDA QMC, “Automotive SPICE: Release Statement,” 2008. [Online]. Available: http://www.automotivespice.com/about/.

[69] Kugler Maag CIE, “Automotive Spice 3.0 - Pocket Guide (Extended HIS Scope),” 2015.

[70] VDA QMC, “Automotive Spice - Process Reference Model & Process Assessment Model 3.0,” 2015.

[71] “Bosch Car Multimedia PT.” [Online]. Available: http://www.bosch.pt/pt/pt/our_company_10/business_sectors_and_divisions_10/car_multimedia_7/car-multimedia.html.

[72] Software Freedom Conservancy, “Git – distributed-is-the-new-centralized,” 2016. [Online]. Available: https://git-scm.com/.

[73] G2 Crowd, “Version Control Systems in 2016,” 2016. [Online]. Available: https://www.g2crowd.com/categories/version-control-systems?order=survey_responses_count.

[74] Jenkins, “Jenkins - Build great things at any scale,” 2016. [Online]. Available: https://jenkins.io/.

[75] I. GitHub, “GitHub - How people build software,” *2016*. [Online]. Available: https://github.com/.

[76] Atlassian, “Bitbucket | The Git solution for professional teams,” *2016*. [Online]. Available: https://bitbucket.org/.

[77] R. Pichler, *Agile Product Management with Scrum: Creating Products that Customers Love*. 2010.

[78] Segue Technologies, “What Characteristics Make Good Agile Acceptance Criteria?,” 2015. [Online]. Available: http://www.seguetech.com/what-characteristics-make-good-agile-acceptance-criteria/.

[79] B. B. Guzina and R. Y. S. Pak, “MoSCoW Prioritisation,” *Science (80-.).*, vol. 33, no. 7, pp. 1005–1021, 1996.

[80] J. Shepley, “Done Agile, Done Right.” [Online]. Available: http://www.aspe-it.com/blog/2013/sharepoint-done-agile-done-right/. [Accessed: 07-Dec-2015].

[81] L. Crispin, J. Gregory, A. P. Lead, O. Mentor, and I. K. Services, *Agile Testing: A Practical Guide*

for Testers and Agile Teams. 2009.

[82] M. McLaughlin, “What is Agile Methodology?” [Online]. Available: https://www.versionone.com/agile-101/agile-methodologies/.

[83] SoftwaySolutions, “There Might Be Scrum-thing To This,” 2012. [Online]. Available: https://www.softwaysolutions.com/blog/might-scrum-thing/.

[84] D. Rawsthorne and D. Shimp, “Scrum in a Nutshell - a Primer.” [Online]. Available: http://agileatlas.org/articles/item/scrum-in-a-nutshell.

[85] Mountain Goat Software, “Release Burndown Chart.” [Online]. Available: https://www.mountaingoatsoftware.com/agile/scrum/release-burndown.

[86] M. Tarnowski, “CMMI — Capability Maturity Model Integration,” 2014. [Online]. Available: http://www.plays-in-business.com/cmmi-capability-maturity-model-integration/.

Appendix I. Principles behind the Agile Manifesto

"We follow these principles:

Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.

Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.

Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.

Business people and developers must work together daily throughout the project.

Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.

The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.

Working software is the primary measure of progress.

Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.

Continuous attention to technical excellence and good design enhances agility.

Simplicity–the art of maximizing the amount of work not done–is essential.

The best architectures, requirements, and designs emerge from self-organizing teams.

At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behaviour accordingly." [21]

Appendix II. CMMI: Continuous Representation

Continuous representation uses a four level rating. Each level has assigned a generic goal, which is achieved by performing the correspondent generic practices. The table below lists the required practices for every capability level.

Level	Generic Goal	Generic Practices
0 – Incomplete	No generic goal	No generic practices
1 – Performed	GG 1. Achieve Specific Goals	GP 1.1. Perform Specific Practices
2 – Managed	GG 2. Institutionalize a Managed Process	GP 2.1. Establish an Organizational Policy GP 2.2. Plan the Process GP 2.3. Provide Resources GP 2.4. Assign Responsibility GP 2.5. Train People GP 2.6. Control Work Products GP 2.7. Identify and Involve Relevant Stakeholders GP 2.8. Monitor and Control the Process GP 2.9. Objectively Evaluate Adherence GP 2.10. Review Status with Higher Level Management
3 – Defined	GG 3. Institutionalize a Defined Process	GP 3.1. Establish a Defined Process GP 3.2. Collect Process Related Experiences

Table 6 - CMMI Capability levels

Appendix III. CMMI: Staged Representation

Staged representation is represented through five maturity levels, which have assigned a predefined set of process areas. The entire group of process areas must be performed in order to achieve the correspondent level. The table below lists all mature levels and its covered process areas.

Level	Focus	Abbr	Process Area
1 – Initial	Process is informal and ad-hoc		
2 – Managed	Basic Project Management	CM	Configuration Management
		MA	Measurement and Analysis
		PPQA	Process and Product Quality Assurance
		PMC	Project Monitoring and Control
		PP	Project Planning
		REQM	Requirements Management
		SAM	Supplier Agreement Management
3 – Performed	Process Standardization	DAR	Decision Analysis and Resolution
		IPM	Integrated Project Management
		OPD	Organizational Process Definition
		OPF	Organizational Process Focus
		OT	Organizational Training
		PI	Product Integration
		RD	Requirements Development
		RSKM	Risk Management
		TS	Technical Solution
		VAL	Validation
		VER	Verification
4 – Managed	Quantitatively Managed	QPM	Quantitative Project Management
		OPP	Organizational Process Performance
5 – Defined	Continuous Process Improvement	CAR	Causal Analysis and Resolution
		OPM	Organizational Performance Management

Table 7 - CMMI Maturity levels

Appendix IV. ASPICE: Process Attributes

In order to measure the extent of achievement of a process attribute within a capability level, ASPICE defined a four point rating scale named NPLF: Not achieved, Partially achieved, Largely achieved, and Fully achieved. Stages are based on repeatability and results of the implemented process attribute, as presented by the following table.

Abbr.	Designation	Description	Achievement
N	Not Achieved	Lacks evidence of achievement of the defined attribute in the assessed process.	≤ 15%
P	Partially Achieved	Although some aspects may be unpredictable, there is some evidence of a systematic approach and a relative achievement of the process attribute.	16 to ≤ 50%
L	Largely Achieved	Besides the repeatability, evidences show a significant achievement of the defined attribute. However, performance of the process may be affected by particular weaknesses.	51 to ≤ 85%
F	Fully Achieved	There are clear evidences of a complete and systematic approach, leading to a full achievement of the process attribute. No significant weaknesses are revealed across the defined organizational unit.	86 to ≤ 100%

Table 8 - ASPICE Rating scale

The correspondence between the consistency of the process attribute and the stages of NPLF scale considers two groups of indicators: process performance indicators, and process capability indicators. The first category, performance indicators, specify the extent of fulfilment of the process outcomes, including base practices and work products. In other hand, the process capability indicators, target the fulfilment of process attribute achievements, namely the generic practices and generic resources. Each process attribute has distinct target indicators, described in detail on the official ASPICE PAM [70].

Finally, to achieve a specific capability level, all its process attributes must be at least Largely achieved, while the process attributes of lower levels must be Fully achieved. For example, a process would only reach capability level three if the process attributes of lower levels were all Fully achieved, and the process attributes of level three at a minimum Largely achieved.

A complete match between capability levels, and required process attributes and its achievement is provided on the table bellow.

Capability Level	Process Attributes		Achievement
Level 1	PA 1.1	Process Performance	Largely/Fully Achieved
Level 2	PA 1.1	Process Performance	Fully Achieved
	PA 2.1	Performance Management	Largely/Fully Achieved
	PA 2.2	Work Product Management	Largely/Fully Achieved
Level 3	PA 1.1	Process Performance	Fully Achieved
	PA 2.1	Performance Management	Fully Achieved
	PA 2.2	Work Product Management	Fully Achieved
	PA 3.1	Process Definition	Largely/Fully Achieved
	PA 3.2	Process Deployment	Largely/Fully Achieved
Level 4	PA 1.1	Process Performance	Fully Achieved
	PA 2.1	Performance Management	Fully Achieved
	PA 2.2	Work Product Management	Fully Achieved
	PA 3.1	Process Definition	Fully Achieved
	PA 3.2	Process Deployment	Fully Achieved
	PA 4.1	Process Measurement	Largely/Fully Achieved
	PA 4.2	Process Control	Largely/Fully Achieved
Level 5	PA 1.1	Process Performance	Fully Achieved
	PA 2.1	Performance Management	Fully Achieved
	PA 2.2	Work Product Management	Fully Achieved
	PA 3.1	Process Definition	Fully Achieved
	PA 3.2	Process Deployment	Fully Achieved
	PA 4.1	Process Measurement	Fully Achieved
	PA 4.2	Process Control	Fully Achieved
	PA 5.1	Process Innovation	Largely/Fully Achieved
	PA 5.2	Continuous Optimization	Largely/Fully Achieved

Table 9 - ASPICE Capability levels

Appendix V. Bosch Innovcar: Project Calendar

Calendar		Iteration	Monday	Tuesday	Wednesday	Thursday	Friday
April	4-8	Sprint #A	Planning	Daily Scrum	Project Official Kick Off	Daily Scrum	Daily Scrum
	11-15		Daily Scrum	Daily Scrum	Bosch Core Team Meeting	Daily Scrum	Review Retrospective
	18-22	Sprint #B	Planning	Daily Scrum	Bosch Core Team Meeting	Daily Scrum	Daily Scrum
	25-29		Daily Scrum	Daily Scrum	Bosch Core Team Meeting	Daily Scrum	Review Retrospective
May	2-6	Release #A			Workshop: Defining the cockpit vision		
	9-13	Sprint #C	Planning	Daily Scrum	Coordination Report Meeting	Daily Scrum	Daily Scrum
	16-20		Daily Scrum	Daily Scrum	Bosch Core Team Meeting	Daily Scrum	Review Retrospective
	23-27	Sprint #D	Planning	Daily Scrum	Bosch Core Team Meeting	Daily Scrum	Daily Scrum
	30-3		Daily Scrum	Daily Scrum	DSM Concept Meeting	Daily Scrum	Review Retrospective
June	6-10	Release #B					
	13-17	Sprint #E	Planning	Daily Scrum	Bosch Core Team Meeting	Daily Scrum	Daily Scrum
	14-20		Daily Scrum	Daily Scrum	Bosch Core Team Meeting	Daily Scrum	Review Retrospective
	27-1	Transitioning Sprint			Toolchain Integration Brainstorming on framework capabilities and initial concepts		
July	4-8	Sprint #1	Planning	Daily Scrum	Bosch Core Team Meeting	Daily Scrum	Daily Scrum
	11-15		Daily Scrum	Daily Scrum	Bosch Core Team Meeting	Daily Scrum	Review Retrospective
	18-22	Sprint #2	Planning	Daily Scrum	Coordination Report Meeting	Daily Scrum	Daily Scrum
	25-29		Daily Scrum	Daily Scrum	Bosch Core Team Meeting	Daily Scrum	Review Retrospective
August	1-5	Release #1					
	29-2	Office Work					
September	5-9	Sprint #3	Planning	Daily Scrum	Core Team Bosch Meeting	Daily Scrum	Daily Scrum
	12-16		Daily Scrum	Daily Scrum	Coordination Report Meeting	Daily Scrum	Review Retrospective
	19-23	Sprint #4	Planning	Daily Scrum	Core Team Bosch Meeting	Daily Scrum	Daily Scrum
	26-30		Daily Scrum	Daily Scrum	Core Team Bosch Meeting	Daily Scrum	Review Retrospective
October	3-7	Release #2			Team Restructuration		
	10-14	Sprint #5	Planning	Daily Scrum	WP Interviews Activity #1	Daily Scrum	Daily Scrum
	17-21		Daily Scrum	Daily Scrum	WP Interviews Activity #1	Daily Scrum	Review Retrospective
	24-28	Sprint #6	Planning	Daily Scrum	WP Interviews Activity #1	Daily Scrum	Daily Scrum
November	31-4		Daily Scrum	Daily Scrum	WP Interviews Activity #1	Daily Scrum	Review Retrospective
	7-11	Release #3			Concepts Demonstration to Bosch & WP Owners		

Table 10 - Bosch innovcar process plan

Appendix VI. CMMI: Matching and Appraisal

Goal	Specific Practices	Rate	Accomplishment
Level 2 Generic Goals			
Institutionalize a Managed Process Process implementation complies with policy, produces expected outputs, and is monitored according to its purpose	Establish an organizational policy	F	✓ Process elements are clearly defined, as practices, roles, and standards. ✓ Resources and tools are available, or under the acquisition process. ✓ Team members are trained according to their roles, and are given authority to perform their assigned responsibilities. ✓ Stakeholders constitute a fundamental part of the process, as they actively participate on scrum events. ✓ Processes are continuously monitored and improved through regular inspections, daily scrums, retrospectives, and coordination meetings.
	Plan the process	F	
	Provide resources	F	
	Assign responsibility	F	
	Train people	F	
	Control work products	L	
	Identify and involve relevant stakeholders	F	
	Monitor and control the process	F	
	Objectively Evaluate adherence	L	
	Review status with higher level management	P	
CM – Configuration Management			
Establish Baselines Configuration of the work products that compose baselines	Identify configuration items	F	✓ Required documentation is previously specified on project charter. ✓ Configuration tools for requirements elicitation, version control, and code integration. ✓ Numbered user stories continuously define baselines for development.
	Establish a configuration management system	L	
	Create or release baselines	F	

Goal	Specific Practices	Rate	Accomplishment
Track and Control Changes Practices to support eventual modifications to established baselines	Track change requests	N	✓ Eventual changes are discussed, as its impact on the project. ✓ Specific team members are responsible for changing and updating configuration systems.
	Control changes to configuration systems	F	
Establish Integrity Ensure the consistency of the established baselines along the modifications	Establish configuration management records	L	✓ Configuration items are tracked along the sprint, and inspected on daily scrums.
	Perform configuration audits	F	
PMC – Project Monitoring and Control			
Monitor Project Against Plan Tracking of actual performance and progress and contrast with established plan	Monitor project planning parameters	F	✓ Scrum board enable the tracking of ongoing work and status of individual tasks. ✓ Burndown charts provide an indication of the product left to complete and the needed effort to achieve it. ✓ Stakeholders are included on review meetings. ✓ Risks are evaluated and its solution is planned through coordination meetings. ✓ Besides the daily inspection of development, the overall progress is tracked for three-month milestones.
	Monitor commitments	F	
	Monitor project risks	F	
	Monitor data management	L	
	Monitor stakeholder involvement	F	
	Conduct progress reviews	F	
	Conduct milestone reviews	F	
Manage Corrective Actions to Closure Action is managed and taken when performance or results deviate from the plan	Analyse issues	F	✓ Daily scrums enable to identify impediments, which may be instantly addressed. ✓ Reviews and retrospectives promote discussion, and help finding possible improvements.
	Take corrective action	F	
	Manage corrective action	P	

Goal	Specific Practices	Rate	Accomplishment
REQM – Requirements Management			
Manage Requirements Develop and sustain a measurement capability to support management information needs	Obtain understanding of requirements	F	✓ Requirements are managed and detailed through the items of Product Backlog. ✓ Team commitment is achieved during the planning meetings. ✓ Backlog is open to change and its requirements, which are immediately updated on the next planning. ✓ Requirements levels as Epics, Stories, and Concepts enable their organization and traceability. ✓ Reviews provide an opportunity to inspect the work, and compare with requirements.
	Obtain commitment to requirements	F	
	Manage requirements change	F	
	Maintain bidirectional traceability of requirements	L	
	Identify inconsistencies between project work and requirements	F	
SAM – Supplier Agreement Management			
Establish Supplier Agreements Contracts are settled and maintained	Determine acquisition type	L	✓ Acquisitions are characterized according to type and value. ✓ Considering metrics as availability and purchase period, a group of preferred suppliers is established.
	Select suppliers	P	
	Establish supplier agreement	N	
Satisfy Supplier Agreements Contracts are fulfilled by both project and supplier	Execute the supplier agreement	N	✓ Compatibility of the acquired products is tested before its inclusion on the project. ✓ Formation, support and additional material is provided to deploy the products.
	Accept the acquired product	L	
	Transition products	F	

Goal	Specific Practices	Rate	Accomplishment
MA – Management and Analysis			
Align Measurement and Analysis Activities Measurement practices and variables are allied with information needs and objectives	Establish measurement objectives	P	✓ Management defines the variables to measure as velocity, features, or story points. ✓ Outputs include statistic numbers and burndown charts. ✓ Procedures are defined according to measurement objectives and supported by management tools.
	Specify measures	F	
	Specify data collection and storage procedures	F	
	Specify analysis and procedures	F	
Provide Measurement Results Results shall be presented, as a help to monitor performance, inform management and enhance technical decisions	Collect measurement data	F	✓ Performance data gathered along the sprint on daily scrums. ✓ Measured data is always available for the team, and exposed to stakeholders on review meetings. ✓ Results are analysed and discussed on retrospective meetings.
	Analyse measurement data	L	
	Store data and results	F	
	Communicate results	F	
PPQA– Process and Product Quality Assurance			
Objectively Evaluate Processes and Work Products Compliance of project elements to standards and procedures is evaluated	Objectively evaluate processes	L	✓ Process is continuously enhanced by the scrum master, and improved through management and team retrospectives. ✓ Integrity of developed product is assured by automated tools. ✓ Acceptance tests determine the conformity with established requirements.
	Objectively evaluate work products and services	F	
Provide Objective Insight Noncompliance issues are objectively tracked, communicated, and resolved	Communicate and ensure resolution for noncompliance issues	F	✓ Product is inspected by management and stakeholders on review meetings. ✓ Identified issues are discussed, and included on the next planning.
	Establish records	P	

Goal	Specific Practices	Rate	Accomplishment
PP– Project Planning			
Establish Estimates Maintenance of planning parameters according to project objectives	Estimate project scope	F	✓ Division and planning of project into modular work packages, which are planned and properly described. ✓ Gates and milestones defined for each work package. ✓ Scrum lifecycle defines iterations and stages of development. ✓ Effort and cost are estimated for iteration.
	Establish estimates of work product and task attributes	L	
	Define project lifecycle	F	
	Determine estimates of effort and cost	F	
Develop Project Plan A formal, approved document used to manage and control the execution of the project and the fulfilment of its requirements	Establish the budget and schedule	F	✓ Project charter defines the budget, deliverables, and overall scope of the project. ✓ Needs and risks are continuously tracked along the sprints, and reported to coordination when confirmed. ✓ Stakeholders are included and perform an important role on review ceremonies. ✓ Every release comprises a development plan. Management issues as risks are planned separately, on coordination meetings.
	Identify project risks	F	
	Plan for data management	F	
	Plan for project resources	F	
	Plan for needed knowledge and skills	L	
	Plan for stakeholder involvement	F	
	Establish the project plan	F	
Obtain Commitment to Plan Plan effectiveness demands commitment by those responsible for its implementation and support	Review plans that affect the project	F	✓ Sprint planning meeting establishes a short-term plan for the current iteration. ✓ Daily scrum ensures continuous inspection and compliance with the committed goals.
	Reconcile work and resource levels	F	
	Obtain plan commitment	F	

Table 11 - CMMI Process matching

Appendix VII. ASPICE: Matching and Appraisal

Process	Base Practices	Rate	Accomplishment
Engineering Process Group			
SYS.2 System Requirements Analysis Transform the requirements into guidelines to design the system	Specify system requirements	F	✓ Requirements are iteratively gathered, and detailed on User Story format. ✓ User Stories are prioritized according to value to customer. ✓ Discussion during grooming and planning meetings identifies dependencies and technical impact. ✓ Verification is achieved through the definition of done and acceptance criteria. ✓ Requirements are agreed on planning meetings, and its consistency is evaluated on the following reviews.
	Structure system requirements	F	
	Analyse system requirements	F	
	Analyse the impact on the operating environment	L	
	Develop verification criteria	F	
	Establish bidirectional traceability	L	
	Ensure consistency	F	
	Communicate agreed system requirements	F	
SYS.3 System Architectural Design Establish a system architectural design, allocate requirements to system elements, and evaluate compliance with defined criteria.	Develop system architectural design	F	✓ Defined general architecture comprising the main components and its interaction. ✓ Systems and their interactions are discussed within multidisciplinary teams. ✓ Research on equivalent systems and different approaches exposes alternative architectures. ✓
	Allocate system requirements	L	
	Define interfaces of system elements	F	
	Describe dynamic behaviour	L	
	Evaluate alternative system architectures	L	

Process	Base Practices	Rate	Accomplishment
SYS.3 System Architectural Design	Establish bidirectional traceability	P	✓ Architecture requirements and evaluation criteria are based on costumer standards. ✓ Architecture is defined with approval of customer, ensuring traceability, consistency, and openness on the architectural strategy.
	Ensure consistency	L	
	Communicate agreed system architectural design	F	
SYS.4 System Integration and Test Ensures the test and integration of system items, as the consistency of the achieved results.	Develop system integration strategy	F	✓ Integration strategy defined according with the committed work, and discussed on planning meetings. ✓ Version control systems organize new integrations for every iteration. ✓ Automated tools evaluate the integrity of the system while new features are integrated. ✓ Compliance with the architectural specification is part of criteria for integration, ensuring its consistency. ✓ Verification is ensured on functional levels. ✓ Integration results are reported and discussed along the sprint, and communicated on daily scrums.
	Include regression test strategy	N	
	Develop specification for system integration test	P	
	Integrate system items	P	
	Select test cases	N	
	Perform system integration test	P	
	Establish bidirectional traceability	N	
	Ensure consistency	L	
	Summarize and communicate results	F	
SYS.5 System Qualification Test Validates the integrated system and evaluates its compliance with the requirements.	Develop system qualification test strategy	L	✓ Test strategy comprises both functional and non-functional test cases. ✓ Acceptance tests are defined with customer, and committed with development team.
	Develop specification for system qualification test	L	
	Select test cases	P	

Process	Base Practices	Rate	Accomplishment
	Test integrated system	L	✓ Test cases are designed for each user story, and performed with integrated system. ✓ Direct relation between user stories and acceptance tests ensures traceability and consistency with customer requirements. ✓ Results are communicated along the sprint, and attested on review demonstrations.
	Establish bidirectional traceability	F	
	Ensure consistency	F	
	Summarize and communicate results	F	
Management Process Group			
MAN.3 Project Management Identify, establish, and control the activities and resources necessary for a project to produce a product.	Define the scope of work	F	✓ Scope of work is macro defined through its division in work packages. ✓ Project life cycle is divided in gates and milestones. ✓ Since goals are established iteratively, their feasibility is continuously evaluated. ✓ Project activities are weekly monitored, while development is daily accompanied. ✓ Work package gates are estimated, and monitored continuously. ✓ Formations and learning sessions are taken according to development needs. ✓ Regular sessions intend to discuss interfaces between teams. ✓ Progress is continuously appraised, and reported to core team involved.
	Define project life cycle	F	
	Evaluate feasibility of the project	P	
	Define, monitor and adjust project activities	L	
	Determine, monitor and adjust project estimates and resources	L	
	Ensure required skills, knowledge, and experience	L	
	Identify, monitor and adjust project interfaces and agreed commitments	F	
	Define, monitor and adjust project schedule	N	
	Ensure consistency	L	
	Review and report progress of the project	F	

Process	Base Practices	Rate	Accomplishment
Supporting Process Group			
SUP.1 Quality Assurance Ensure that work products and processes comply with predefined previsions, and non-compliances are resolved and further prevented.	Develop a project quality assurance strategy	N	✓ Quality strategy agreed with customer and defined according to innovative nature of the project. ✓ Quality reviews on work products periodically conducted under reviews and release presentations. ✓ Process is continuously appraised and improved through retrospective meetings. ✓ Non-conformances are communicated on daily scrums, and reported to coordination when needed.
	Assure quality of work products	L	
	Assure quality of process activities	F	
	Summarize and communicate quality assurance results.	P	
	Ensure resolution of non-conformances	F	
	Implement an escalation mechanism	L	
SUP.8 Configuration Management Establish, check availability, and maintain the integrity of all work products.	Develop a configuration management strategy	F	✓ Configuration strategy addresses team roles and responsibilities, tools and repositories, and integration procedures. ✓ Branch management strategy developed according to sprints and related user stories. ✓ Eventual modifications and releases are previously discussed within the team. ✓ Team charters, internal code naming conventions, integration procedures, and other work products establish baselines for development. ✓ Configured items are daily tracked, and kept at local repositories.
	Identify configuration items	L	
	Establish a configuration management system	N	
	Establish branch management strategy	F	
	Control modifications and releases	F	
	Establish baselines	P	
	Report configuration status	N	
	Verify the information about configured items	F	
	Manage the storage of configuration items and baselines	F	

Process	Base Practices	Rate	Accomplishment
SUP.9 Problem Resolution Management Certify that problems are identified, analysed, managed and controlled to resolution.	Develop a problem resolution strategy	F	✓ Problems reported during daily scrums are immediately addressed by the scrum master. ✓ When needed, problems are tracked and reported to core team involving customers and project partners. ✓ Status and impact of the problem are addressed on an *Open Points List*. ✓ If the problem can be solved within the development team, its resolution is immediately initiated. ✓ Problems are continuously tracked through regular core team meetings.
	Identify and record the problem	N	
	Record the status of problems	P	
	Determine cause and impact of the problem	P	
	Authorize urgent resolution action	F	
	Raise alert notifications	F	
	Initiate problem resolution	L	
	Track problems to closure	P	
	Analyse problem trends	N	
SUP.10 Change Request Management Certify that change requests are tracked and implemented.	Change request management strategy	N	✓ Change requests are discussed on core team meetings with customers and project partners, and afterwards discussed within the development team. ✓ Before implementing any change request, risks, benefits, and impact on project are duly evaluated. ✓ Change requests are followed by an experimental period, in which the new approach is tested. ✓ Changes are tracked until implementation, and correspondent feedback is provided to customer and other interested parts.
	Identify and record the change requests	P	
	Record the status of change requests	N	
	Analyse and assess change requests	F	
	Approve and review change requests	L	
	Track change requests to closure	F	
	Establish bidirectional traceability	F	

Process	Base Practices	Rate	Accomplishment
Supporting Process Group			
SWE.1 Software Requirements Analysis Clarify the requirements of software related part of the system.	Specify software requirements	F	✓ Requirements are refined with customer, and expressed by the form of user stories. ✓ User stories are grouped and prioritized according to customer needs. ✓ User stories are clarified and discussed on grooming sessions, addressing the prospective impact on project. ✓ Acceptance criteria provide detail on functional, non-functional, and performance requirements for each feature. ✓ Requirements are estimated and committed on planning meetings. ✓ Backlog ensures traceability, while review meetings provide an opportunity to inspect consistency and communicate results.
	Structure software requirements	F	
	Analyse software requirements	F	
	Analyse the impact on the operating environment	F	
	Develop verification criteria	F	
	Establish bidirectional traceability	F	
	Ensure consistency	F	
	Communicate agreed software requirements	F	
SWE.2 Software Architectural Design Establish an architectural design and identify how software requirements shall be allocated.	Develop software architectural design	P	✓ Overall architecture is defined according to system work package requirements. ✓ Software requirements are gathered iteratively and resultant interfaces are discussed along the sprints. ✓ Despite not being planned, dynamic behaviour continuously evaluated.
	Allocate software requirements	L	
	Define interfaces of software elements	F	
	Describe dynamic behaviour	P	
	Define resource consumption objectives	N	
	Establish bidirectional traceability	P	

Process	Base Practices	Rate	Accomplishment
SWE.2 Software Architectural Design	Ensure consistency	F	✓ Architectural weaknesses and inconsistencies are tracked along the sprints, and reported to the team on daily scrums.
	Communicate agreed architectural design	F	
SWE.3 Software Detailed Design and Unit Construction Produce and provide a detailed design for the software units.	Develop software detailed design	P	✓ Software functional and non-functional behaviour specified through acceptance criteria and discussed with design team. ✓ Software design is implicit on defined tasks for each feature. ✓ Interfaces are discussed and evaluated while planning the development. ✓ Relation between criteria, tasks, and user stories establishes traceability between requirements and developed units. ✓ Consistency is continuously inspected and reported through daily scrums.
	Define interfaces of software units	P	
	Describe dynamic behaviour	F	
	Evaluate software detailed design	N	
	Establish bidirectional traceability	L	
	Ensure consistency	F	
	Communicate agreed software detailed design	F	
	Develop software units	L	
SWE.4 Software Unit Verification Verify software units to provide evidence for compliance between software units and correspondent requirements and design.	Develop software unit verification strategy	P	✓ Verification strategy includes acceptance tests and code reviews. ✓ User stories constitute software units, which are tested separately. ✓ User stories are tested continuously along the sprint. ✓ Dependencies between user stories and correspondent acceptance tests ensure traceability. ✓ Relation between user stories and designed tests ensures consistency.
	Develop criteria for unit verification	L	
	Perform verification of software units	L	
	Test software units	N	
	Establish bidirectional traceability	P	
	Ensure consistency	L	

Process	Base Practices	Rate	Accomplishment
	Summarize and communicate results	F	✓ Team discusses results on review meetings.
SWE.5 Software Integration and Integration Test Integrate software units into the global system, ensuring consistency and compliance with both software and architectural designs.	Develop software integration strategy	F	✓ Integration steps discussed on planning meetings, and organized according to user story prioritization. ✓ Automated tools and scripts as Jenkins support the continuous integration practice. ✓ Results and logs resultant from integration episodes are provided by automated tools. ✓ Integration results are placed on local repositories, and communicated within the team during daily meetings.
	Develop software integration test strategy including regression test	N	
	Specification for software integration test	N	
	Integrate software units and software items	F	
	Select test cases	N	
	Perform software integration test	P	
	Establish bidirectional traceability	N	
	Ensure consistency	N	
	Summarize and communicate results	F	
SWE.6 Software Qualification Test Ensure the integrated software is tested and compliant with established requirements.	Develop software qualification test strategy including regression test	P	✓ Testing strategies are discussed iteratively, considering the committed work for the ongoing sprint. ✓ Qualification criteria is specified for each user story. ✓ Software is tested continuously by the pace of integration. ✓ Consistency and traceability of requirements are ensured through relation between user stories and acceptance criteria. ✓ Results are analysed on review meetings, and shared within the team.
	Specification for software qualification test	L	
	Select test cases	N	
	Test integrated software	L	
	Ensure consistency and bidirectional traceability	P	
	Summarize and communicate results	F	

Process	Base Practices	Rate	Accomplishment
Supporting Process Group			
ACQ.4 Supplier Monitoring Track and assess the performance of the supplier against agreed requirements.	Agree and maintain joint processes	P	✓ Supplier agreements include support, documentation, and training if needed. ✓ Contacts are mainly performed through intermediary companies. ✓ Supplier follows the development until the acquired product becomes established.
	Exchange all agreed information	F	
	Review technical development with the supplier	L	
	Review progress of the supplier	N	
	Act to correct deviations	F	

Table 12 - ASPICE Process matching