Présentée et soutenue par :

Mme VALENTINA MARIA LLAMAS ZOGBI

le mardi 14 novembre 2017

Titre :

Towards an agile methodology for industrial problem solving / Vers une méthodologie agile pour la résolution de problèmes industriels

TOWARDS AN AGILE METHODOLOGY FOR INDUSTRIAL PROBLEM SOLVING

ACKNOWLEDGMENTS

I would like these paragraphs to express my deepest and sincere gratitude to all the people that, with their advice and support, collaborated with the accomplishment of the present work.

First and foremost, I would like to thank both of my academic advisors, Laurent Geneste and Thierry Coudert. Thanks for your support and guidance through these last three years. I really appreciate all your contributions of time and ideas that truly helped and motivated me. Thanks for your precious advice and enormous knowledge.

I would also like to thank my industrial supervisor, Juan Romero. Thank you for providing me the opportunity of joining this collaborative research. I am especially grateful for the continuous support and encouragement both on professional and personal levels.

Further, I would like to acknowledge Marie-Anne Le Dain and Samir Lamouri for taking the time to appraise my work and for their very interesting suggestions, and to Maurice Pillet for accepting to be the president of the jury.

Thanks to Aymeric de Valroger and Jean-Claude Simonet, president and associate director of Axsens bte, for welcoming me to the company and mostly for your insightful comments and guidance. I also want to thank all my colleagues from Axsens bte and from ENIT-LGP. I would like to recognize Sofia, for she accompanied me during this last year with positive energy and motivation.

I thank my dearest friends for your friendship and support, most times in the distance. I am mostly grateful to my family for their unconditional support during my entire career. Thanks to my sister, my brothers and especially to my parents, a source of inspiration and my example to follow in all levels of life. Thanks to my grandparents for their excellent advice and continuous incentive.

My deepest and truly acknowledgment goes to my loving, encouraging and patient Luciano. I am grateful for your faithful support and inspirational words. Thanks for accompanying me into this project and for all those to come.

TABLE OF CONTENTS

List of Figures

List of Tables

1. INTRODUCTION

This PhD thesis has been performed in the framework of a long-lasting partnership between the company Axsens-bte and the laboratory LGP-ENIT (Laboratoire Génie de Production – Ecole Nationale d'Ingénieurs de Tarbes). Axsens-bte[1] is a consulting and training firm specialized in the domains of continuous improvement, problem solving, supply chain and industrial methods, among others. In the company, the research was carried out within the department "Research and Methods" (R&M). The R&M department aims at defining innovative and practical solutions for Axsens-bte customers through applied research and collaborative projects. On the other hand, the LGP[2] is the laboratory of the ENIT, an engineering public school integrated in the University of Toulouse and linked to the "Institut National Polytechnique de Toulouse" (INPT). This research was conducted within the "Systèmes Décisionnels et Cognitifs" (SDC) team, one of the four research teams of the laboratory. The domains of research of the SDC team are: Experiences, Knowledge and competencies (ECC from French); and uncertainty, risk and decisions (IRD from French). Furthermore, another PhD thesis has already been carried out between Axsens-bte and ENIT: "Collaborative problem solving within supply chains: general framework, process and methodology" (Romero Bejarano, 2013), and a third PhD project has been launched in the framework of this partnership in 2016.

Moreover, this work has been performed through a specific doctoral program based on the public-private partnership framework, called CIFRE (from French, "Conventions Industrielles de Formation par la REcherche") which supported by the French Ministry of Higher Education and Research.

In such a context, both Axsens-bte and LGP-ENIT have been dealing with strong industrial issues for many years. Organisation, modelling, optimization, methods and tools are needed to face these issues. That is why Axsens-bte and LGP-ENIT are collaborating via the CIFRE framework to provide new paradigms.

In a highly changing and unstable environment, modern companies need to be ready to capture market requirements and to propose adequate solutions. Then, organisations should constantly adapt their continuous improvement strategies to face changes in the marketplace. More specifically, industrial problem solving is one of the key activities that support continuous improvement approaches. In order to provide improvement and innovation in the problem solving domain, the partnership Axsens-bte/LGP has worked together for many years. The first PhD thesis above referenced, and an open and free software to support industrial problem solving, ProWhy[3], are examples of this collaborative work which aims at satisfying industrial needs.

[1] http://axsens.com/
[2] http://www.lgp.enit.fr/fr/lgp.html
[3] http://prowhy.org/

Moreover, the adaptation of nowadays companies to face changes in their environment can be achieved through the revision and adjustment of their business processes. The business processes context is then very important for the organisation of companies. Some definitions and insights on the topic which has been identified in the literature are presented next.

1.1 Business processes

A process is "*a specific ordering of work activities across time and place, with a beginning, an end, and clearly identified inputs and outputs*" (Davenport, 1993). Moreover, a business process is "*a collection of tasks and activities (business operations and actions) consisting of employees, materials, machines, systems and methods that are being structured in such a way as to design, create, and deliver a product or a service to the customer*" (von Scheel et al., 2015). The concept of *process* has evolved until these days, as described in (von Scheel et al., 2015). The study presented by von Scheel et al. lists influencing people that modified the vision of processes, and their propositions (methods or approaches) which brought the progress and evolution of processes. Authors define four major phases of process evolution. The first phase goes from Sun Tzu (544 BC), with his book on military strategy describing the use of process activities to fulfil specific goals, to Henry Ford's mass production. The second phase, focused on workers and process optimization, goes from Henry Laurence Gantt with his well-known Gantt chart used to present phases and activities in a project over time, follows with the simplification and optimization of processes until the Toyota Production System (TPS). In the third phase, concepts of process visualization and digitalization are introduced, from the approach to enterprise modelling and enterprise architecture ARIS, through Lean thinking and practices; Business Process Reengineering, Total Quality Management (Duret and Pillet, 2011); until Six sigma, a set of methods and tools for process improvement. The fourth phase is composed of Business Process Management (BPM) and its notation BPMN, which are further developed afterwards.

There exist several definitions of BPM in the literature. "*BPM is a 'process process', it is the process for managing the company processes*" (Debauche and Mégard, 2004). According to (Swenson and von Rosing, 2015) BPM "*is a discipline involving any combination of modelling, automation, execution, control, measurement, and optimization of business activity flows in applicable combination to support enterprise goals, spanning organisational and system boundaries, and involving employees, customers and partners within and beyond the enterprise boundaries*". In more general terms, the BPM approach supports and controls business processes through their analysis, modelling, simulation, documentation and execution, aiming at increasing the benefits of the organisation through the improvement of its effectiveness (Aalst et al., 2003). Moreover, BPM philosophy contains six main phases: the first phase is Analyse (project preparation and blueprint), the second is Design (project realization and design), third, Build (final project preparation), fourth, Deploy/implement (Go live), fifth, Run/Maintain (run processes and govern performance), and finally, Continuous improvement (Continuously optimize and develop processes) (von Rosing et al., 2015a).

Furthermore, in order to model business processes, different techniques exist, such as Business Process Model Notation (BPMN), a standard notation that provides elements to graphically specify business processes. It aims to support BPM through an easily understandable set of symbols in order to allow their comprehension by both business and technical users. It is based on three symbols categories: the model elements (events, activities, gateways), the connecting means (sequences, messages, associations) and the element groups (pools, lanes) (Debauche and Mégard, 2004). Moreover, ArchiMate is a high-level architecture modelling standard. It could be adopted by enterprises operating in a dynamic and complex environment, combined with different low detailed level modelling standards (e.g. BPM, UML) (Gill, 2015).

Over years, a certain level of standardization has been achieved on business processes. Recurrent and static processes (such as the billing process) are, in most cases, performed in different situations following the same steps. Then, such processes can be standardized through the definition of a predefined structured approach, which can be systematically deployed. Business processes need to be standardized and structured in modern companies, for instance, to support Enterprise Resource Planning (ERP) software structured manner of working.

Therefore, some routine or recurrent business processes exist within organisations. Such processes are often performed following a predefined and structured workflow. However, business processes can also be non-structured or "exploratory". Unlike structured processes, in this case, the workflow of the process is completed all along the execution of the process. It means that there is no predefined "path" to follow, then, decision makers do not know how they will perform the process before deploying each step. Structured and non-structured types of generic processes, illustrated in Figure 1, are described next.

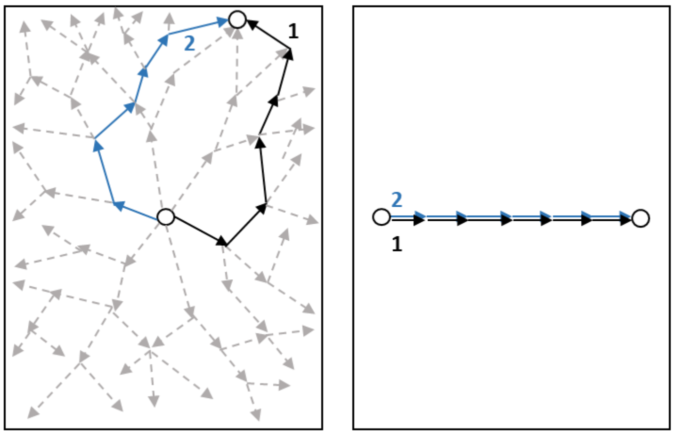


Figure 1a: Exploratory process **Figure 1b**: Structured process

Figure 1: Generic processes structuration

Figure 1a represents a non-structured or exploratory process (Lechner and Floyd, 2007). This type of process presents a high level of flexibility that allows readjustments through different alternatives. When facing a disturbance, the process can be reconfigured to reach the objectives, for instance, new activities can be added in real time to overcome problems. However, this high level of flexibility involves a low level of formalization since it is quite difficult to define standards for their systematic reuse. Then, the formalization and use of general knowledge is difficult to achieve.

At the opposite, the situation in Figure 1b describes a structured process. The process is formalized as a set of pre-defined activities, allowing its systematic reuse. The advantages are both, that activities can be performed without ambiguities, and that knowledge can be formalized and used to help decision makers. However, in such a type of process, when unexpected events or disturbances occur, it is very difficult to change and to react. Furthermore, when overcoming non-formalized scenarios (i.e. new activities performed out of the standard), the standardized available knowledge can be inadequate.

Then, two extreme situations can be recognized: completely structured and exploratory processes. However, a third case exists which combines both types of processes. There are some business processes that can be structured and, at the same time, contain exploratory parts. Such a combination of both situations could improve business processes by taking into account the advantages of each one of them. More specifically, the third type of process presents a certain level of flexibility that could be used to improve business processes. In our work, this third approach is denominated an *agile process*, represented in Figure 2b.

An agile process is defined through the combination of both previously described extreme situations. Thus, drawbacks from both structured and non-structured processes are taken into account to define an agile process that:

- is sufficiently structured to ensure objectives satisfaction and process efficiency but not over constrained by standards,
- can be reconfigured and adapted to unexpected situations,
- is based on experience feedback principles (i.e. the process is driven by knowledge and experiences reuse and allows learning during its execution).

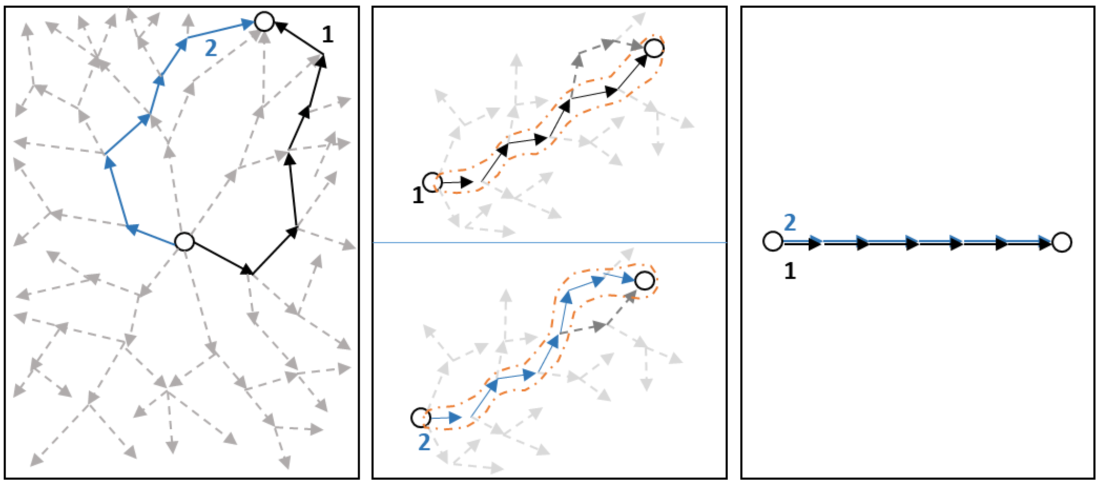


Figure 2a: Exploratory process **Figure 2b**: Agile process **Figure 2c**: Structured process

Figure 2: Three types of generic processes

Then, agile processes combine exploratory and structured processes. In the same way that fully structured and fully exploratory processes exist, there also exist processes that are composed by a combination of both exploratory and structured processes. Such is the case of industrial problem-solving processes, described in the next section.

1.2 Problem-solving processes

Problem Solving is one of the main activities carried out regularly by companies with the purpose of improving quality and achieving continuous improvement. A large number of methods, such as Plan, Do, Check, Act (PDCA), Define, Measure, Analyse, Control (DMAIC), 8 Disciplines (8D) / 9 Steps (9S) exist. These methods are based on specific approaches defining steps to follow in order to solve a problem (problem-solving approach and methods are described in section 2.4). Such predefined approaches are, in most cases, very structured. The success of these methods strongly depends, indeed, on the disciplined approach required to systematically perform the pre-defined resolution workflow that they propose.

In addition to this, they are often applied systematically and regardless of the nature of the problem to solve, the context that surrounds it and the level of skills and evidence available to eradicate it. Therefore, a lack of flexibility to adjust the problem-solving process to deal with simple to complex, interconnected and highly changing context and problems has been observed during the application of these methods in industry.

Furthermore, a typical characteristic of problem-solving processes is that they are defined on a rigid framework. It means that the defined steps in order to solve a problem must be always applied in a predefined manner (which is the principle of a structured process). Nevertheless, there are activities that often do not fit this rigidity concept because of their highly exploratory nature. For instance, a typical exploratory activity is the analysis of root causes. It is a non-structured step due to a high level of uncertainty regarding the specific task to be performed since different tools and techniques exist in order to define root causes.

Then, problem-solving processes constitute an interesting target to define and to apply agility principles since they combine structured and exploratory processes. Taking into consideration the statement regarding business process and more specifically, problem-solving processes, the problematic that this research work addresses is outlined in the next section.

1.3 Problem statement and objectives

Business process context was presented in the first section. Such processes, in most cases, are defined following a structured workflow. Exploratory processes also exist in organisations, where no predefined path is followed during process execution. However, a process can be a combination of structured and exploratory sub-processes or activities, which is the case of problem-solving processes. This thesis work studies a third type of process between both extreme situations, what we call an agile process. Then, the problem addressed in this work can be summarized as:

Problem-solving processes are not sufficiently agile in order to continuously adapt their workflow when facing changes. In most cases, they are not defined according to their context, the nature of the problem to be solved and the possible events (expected or unexpected) that could impact their execution and that could require reconfiguration.

In order to tackle the research problem, two global objectives are defined in our work: The first objective is a methodological one, and the second one regards more technical aspects.

- First objective: The improvement of flexibility and adaptability of problem-solving processes through the incorporation of agile principles. Increasing and adapting the learning of problem-solving processes from previous situations through the incorporation of knowledge reuse principles. Such concepts could be applied to problem-solving processes in order to enable their agility.
- Second objective: Support of existing problem-solving processes through the definition of appropriate algorithms and mechanisms. Such mechanisms could help to increase the agility of the process.

Those global objectives will be further refined with respect to the bibliographic review. Then, at the end of the state of the art section, the global objectives are re-defined.

1.4 Structure of the document

This PhD thesis work intends to define an agile problem-solving process model based on the capitalization and reuse of knowledge and experiences. The document is organised as follows.

The bibliographic study carried out for this research is detailed in chapter 2. It is divided into the three research pillars necessary to define the originality of our work: the domains related to: i) the agility concept, ii) knowledge and experience capitalization and reuse concepts, and iii) the problem-solving approaches and methods. The link between the three research pillars and the concepts that will be considered for our study are presented. This bibliographic review leads to deepen and to refine the objectives of this work

The information model for agility is introduced in chapter 3. The information about the elements of an agile process (activities, decision-making points and versioning system) are detailed. The knowledge capitalization and reuse mechanisms are also detailed and the structure and way of working of the knowledge and experience bases are described. A tagging system enabling the characterization and future retrieval of the process is also introduced, as well as specific indicators that are defined with the purpose of constraining the process.

In chapter 4, the agile lifecycle is described. Such a lifecycle, based on CBR (Case based-reasoning) principles, guides the definition, execution and storage of an agile process. The chapter is divided following the five steps of the agile lifecycle: 1/ Process scope definition, 2/ Experience filtering, 3/ Adaptation of the first version of the process, 4/ Process execution/continuous adaptation, 5/ Storage in the Experience/Knowledge bases.

In chapter 5, an application of the method on an industrial context is introduced. It was conducted at a surface treatment company and it intends to clarify the agile lifecycle through the application of the model and its elements to a real case.

Finally, the last section concludes with a synthesis of the contributions of this work. Also, the identified perspectives and further work related to this research are presented.

2. STATE OF THE ART

2.1 Introduction

This study brings together three research domains in order to define an agile knowledge and experience based process model. First, agility principles and concepts which are intended to provide flexibility and adaptability to process management need to be detailed. Second, in order to define appropriate knowledge and experience capitalization and reuse mechanisms for agile processes, such concepts need to be defined. It has been argued that problem-solving processes, due to their dual (structured / exploratory) nature, will be used as the application of this study. Problem solving is thus, the third research domain that needs to be described for this work.

This chapter outlines the founding principles and key concepts of each one of the three domains in order to clearly identify the concepts that will be used and/or adapted for our model. The three research domains are illustrated in Figure 3.

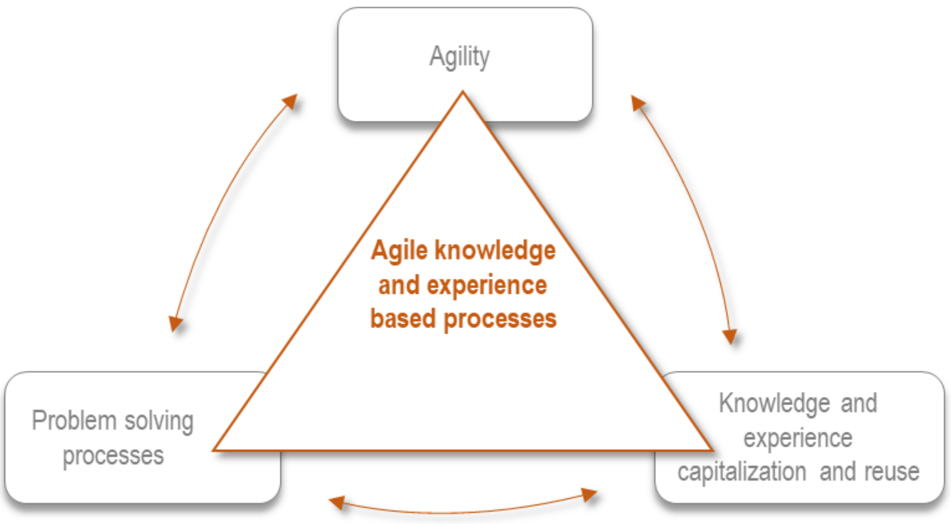

Agility
Agile knowledge and experience based processes
Problem solving processes
Knowledge and experience capitalization and reuse

Figure 3: Three research domains of this study

Section 2.2 outlines agility, a wide studied concept that has been divided into three sub-sections tracing main applications domains: Agile software development methods are described in section 2.2.1, Agile organisations principles are described in section 2.2.2 and Agile business processes and workflows are described in section 2.2.3. In section 2.2.4 a summary of the concepts and characteristics of agility is presented, followed by the definition of the key features of an agile process. In section 2.3 the domain of knowledge and experience capitalization and reuse is presented. The experience feedback approach is introduced in section 2.3.2, followed by the link between agility and knowledge and experience capitalization and reuse concepts. Section 2.4 describes problem-solving processes. First, in section 2.4.1 the industrial problem-solving approach is introduced, describing why problem solving is necessary in today's companies and how it is applied. In section 2.4.2 some of the standard problem-solving methods are introduced. In section 2.4.3, the link between agility and problem solving is presented, followed by the link between problem solving and knowledge and experience capitalization and reuse. Finally, in section 2.5, the synthesis of this chapter and the contributions of this work are presented.

2.2 Agility Concepts

The concept of agility has been at the heart of several research works. Many definitions of agility exist for different application domains. According to the Cambridge dictionary, agility is "*the ability to move about quickly and easily*" (Cambridge University Press, 2017). The concept of agility is sometimes used indistinctly with the concept of flexibility. Flexibility refers to "*the capability of an organisation to move from one task to another quickly and as a routine procedure, with each situation defined ahead of time so that the procedures needed to manage it are in place*" while agility can refer to this definition but it adds its main driver that is "*the ability to respond quickly to unanticipated changes*" (Vokurka and Fliedner, 1998). Furthermore, many authors agree that flexibility is one of agility main characteristics (Sharifi and Zhang, 1999; Yusuf et al., 1999; Lin et al., 2006; Agarwal et al., 2006; Ren et al., 2009). Different *types* of agility are described in the literature: *Customer agility* refers to the incorporation of the customer to the identification and creation of new opportunities for innovation (Raschke, 2010); *partnering agility* allows a firm to exploit opportunities through the access to its extended enterprise network's competencies (Sambamurthy et al., 2003); and *operational agility* ensures that the firm can rapidly change and adapt its processes to face the changes in the environment (Sambamurthy et al., 2003).

Moreover, *agility* is a concept that has been studied in several domains such as agile software development, agile manufacturing, agile enterprise, agile supply chains, business process agility, and agile workflows. Related works corresponding to these domains are synthesized in Table 1.

Domain	**References**
Agile software development methods	(McCauley, 2001; Fowler and Highsmith, 2001; Schwaber, 2004; Lindstrom and Jeffries, 2004; Qumer and Henderson-Sellers, 2008; Tarhan and Yilmaz, 2014)

Agile manufacturing	(Nagel and Dove, 1991; Sharifi and Zhang, 1999; Yusuf et al., 1999; Katayama and Bennett, 1999; Kettunen, 2009)
Agile enterprise	(Dove, 1994; Ren et al., 2009; Gill, 2015)
Agile supply chain	(Christopher, 2000; Lin et al., 2006; Agarwal et al., 2007; Sangari et al., 2015)
Business process agility	(Raschke and David, 2009; Raschke, 2010; Seethamraju and Krishna Sundar, 2013; von Rosing et al., 2015; Battistella et al., 2017)
Agile workflows	(Weber and Wild, 2005; Minor et al., 2014; Bergmann and Gil, 2014)

Table 1: Agility related works

From section 2.2.1 to section 2.2.3 agile concepts from different domains are presented. Finally, the adapted agile key concepts and characteristics from the bibliographic study are presented in section 2.2.4.

2.2.1 Agile software development methods

In the 1990s, traditional software development methods started to fail, representing low success rates of software projects (Tarhan and Yilmaz, 2014). This was caused by a constant technical evolution in the domains of computers, software and communications technology (Lindstrom and Jeffries, 2004). Afterwards, in the late 1990s, agile software development methods started emerging as alternatives to provide improvements to traditional methods (McCauley, 2001).

The **waterfall process** (Royce, 1987) was the first formal software development method. It is still used when project requirements are fixed. It is based on five main practices: System concept, Analysis, Design, Coding and Testing (Qumer and Henderson-Sellers, 2008).

The **spiral model** (Boehm, 1988) for software development is also a well-known traditional method. It may be characterized by five main phases: Project concept, Risk management, Planning, Prototyping and Product engineering (Qumer and Henderson-Sellers, 2008). Predictive methods such as waterfall produce rigid requirement documents to be used in the development lifecycle, impeding any changes in requirements. Even if the spiral model presents many of the waterfall method disadvantages, its incremental development process enables to adapt to unstable requirements (Nuseibeh, 2001).

In 2001, some agile software developers worked together to share practices. Then, the Agile Alliance was created, formally introducing agility through the **Agile Manifesto** (Table 2). The Agile Manifesto lists values and principles common to all agile methods (Lindstrom and Jeffries, 2004).

We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:

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

That is, while there is value in the items on the right, we value the items on the left more.

Table 2: Manifesto for Agile Software Development ("Manifesto for Agile Software Development," 2001)

Two of the authors of the agile manifesto explain that the distinction between the first and the second part of each value (i.e. each item in Table 2) lies at the *heart* of agility (Fowler and Highsmith, 2001).

Moreover, a set of agile principles is included in the agile manifesto (see Appendix 1). According to (McCauley, 2001), the general principles that these agile methods introduce are flexibility and adaptability in order to respond to changes in requirements all over the project.

Despite the values and principles outlined by the agile manifesto, there is no unified definition of an agile software development method. In (Qumer and Henderson-Sellers, 2008), the authors claim that "*a software method is said to be an agile software development method when a method is people focused, communications-oriented, flexible (ready to adapt to expected or unexpected change at any time), speedy (encourages rapid and iterative development of the product in small releases), lean (focuses on shortening timeframe and cost and on improved quality), responsive (reacts appropriately to expected and unexpected changes), and learning (focuses on improvement during and after product development)*".

Based on the principles and values of the Agile Manifesto, several *agile* software development methods have emerged. Two of the most deployed methods are Extreme Programming and Scrum (Tarhan and Yilmaz, 2014). Other well-known agile methodologies

(Qumer and Henderson-Sellers, 2008) are: Feature Driven Development (FDD) (Palmer and Felsing, 2002), Adaptive Software Development (ASD) (Highsmith, 2013), Dynamic Software Development Method (DSDM) (Stapleton, 1997) and Crystal (Abrahamsson et al., 2003). Extreme Programming and Scrum methodologies are detailed next.

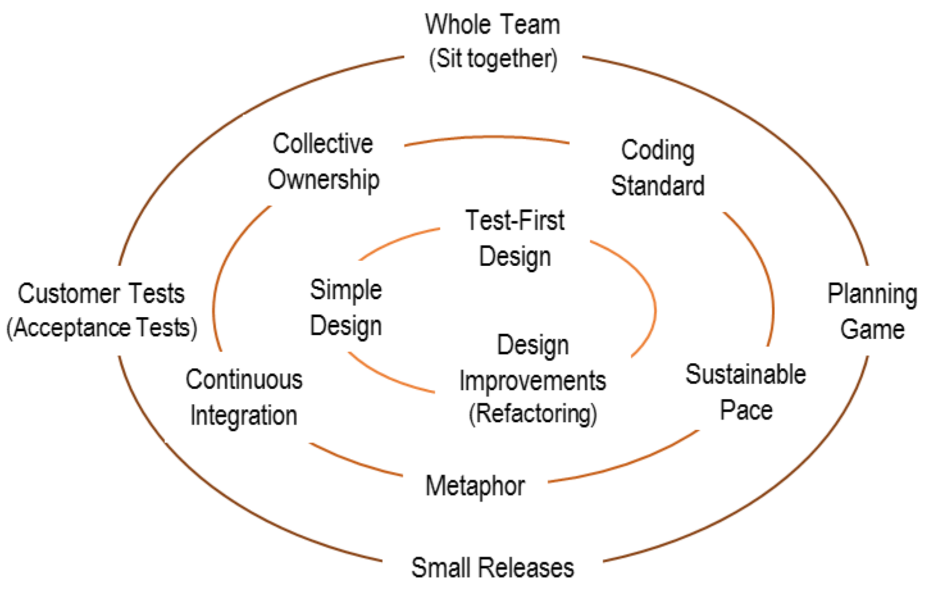


Figure 4: XP practices and cycle of life by (Lindstrom and Jeffries, 2004)

Extreme Programming (XP) is a lightweight and efficient methodology for small-to-medium-sized teams that develop software from highly changing requirements (Beck, 2000). It is focused on iterative and rapid development (Qumer and Henderson-Sellers, 2008). Six roles are included in an XP team: programmer, customer, tester, tracker, coach and manager. This method is based on values of simplicity, communication, feedback and courage, applied through its twelve main practices. XP main practices are illustrated in Figure 4 by (Lindstrom and Jeffries, 2004). In the illustration, the twelve practices are distributed in three circles, from the inside to the outside. It represents: 1) the programmers, 2) the practices that helps the team to collaborate to deliver quality software and 3) the planning cycle that takes place between programmers and customers. Each practice is defined by (Lindstrom and Jeffries, 2004) as follows:

- Whole team: All of the contributors to the project work together as one team. It includes the programmers (developers, analysers, testers, etc.) and the customer.
- Planning game: It consists of defining what will be delivered by the due date and deciding what to do next. It is composed of two phases: First, the Release planning consists of establishing an initial plan for the project based on the desired features and their difficulty. The plan, mostly imprecise, is regularly revised along the project. Second, the Iteration planning is defined considering the features demanded by the customer for the next two weeks (iterations duration is usually of 2 weeks) and the

estimated workload to perform it along with the previous iteration outputs.

- Customer tests: Acceptance tests are defined by the customer to ensure that features are working. Tests are built and implemented by the team.
- Small releases: Integrated tested software is released with every iteration to the customer who can release it to the end user. Also, programmers frequently release software to their own end users.
- Simple design: A key principle of XP is to create and maintain, during tests and improvements, a simple design.
- Pair programming: Programmers work in pairs to ensure that all the software is reviewed by at least one person. This practice helps to improve the code and knowledge sharing among the team.
- Test-driven development: All the tests are run all along the project. This continuous feedback helps to improve the design of the software.
- Design improvement: A continuous design improvement process called refactoring is used. It ensures the removal of duplication and increases the consistance of the code.
- Continuous integration: The system is fully integrated all along the project. It avoids serious problems such as the non-detection of problems during integration due to non-integrated tests.
- Collective code ownership: All programmers are familiar with the code and they can all add improvements at any moment.
- Coding standard: A common standard is used to code so that all programmers are familiar with it.
- Metaphor: The metaphor is a *description* or a *vision* of the system, used to ensure that everyone understands how the software works.
- Sustainable pace: Teams work in a pace that can be sustained indefinitely in order to ensure efficiency.

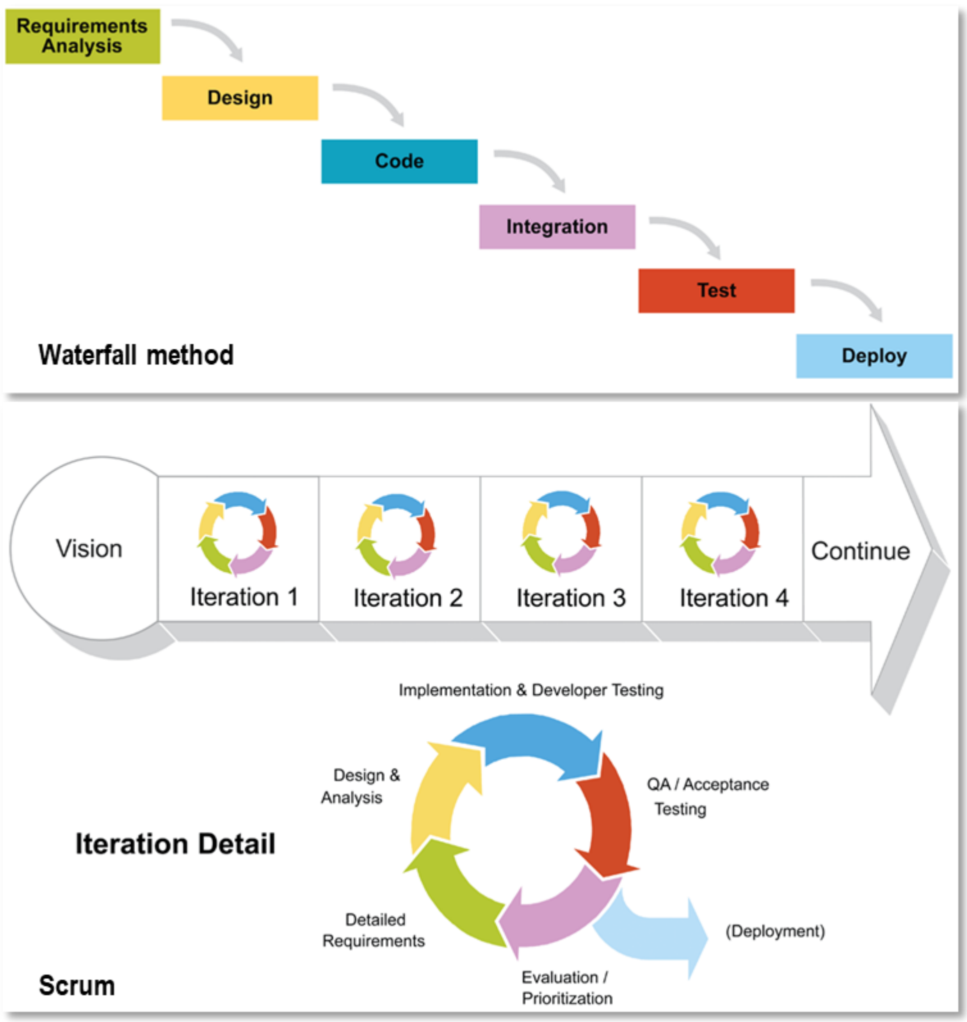


Figure 5: Traditional waterfall development and Scrum development by (James, 2010)

Scrum is "*a framework within which people can address complex adaptive problems, while productively and creatively delivering products of the highest possible value*" (Schwaber and Sutherland, 2001), developed by Ken Schwaber and Jeff Sutherland and applied since the 1990s. Scrum searches to introduce flexibility, adaptability and productivity into the system (Schwaber and Beedle, 2002).

In Figure 5, the difference between a waterfall method and Scrum is shown by (James, 2010). In the waterfall method, requirements need to be perfectly understood from the beginning and errors are propagated through phases. In the Scrum method, on the other hand, all development activities are blended within each iteration.

Scrum is an agile method grounded on the values of commitment, courage, focus, openness and respect (Schwaber and Beedle, 2002). Scrum is based on a 'skeleton' composed by a cycle of iterations and inspections working as follows: the team, based on the initial requirements, decides what to develop in the iteration or *sprint*, works by self-managing to accomplish the objective of the iteration and, when finished, presents the increment of functionality to stakeholders for inspection and, if necessary, adjustments for the next sprint (Schwaber, 2004). To implement this incremental cycle, Scrum sets a structure of *roles*, *meetings* and *artifacts*. Each component of Scrum is detailed next.

The Scrum team, a self-organized and cross-functional group of people, allows flexibility, creativity and productivity optimization into the project. Three roles are defined within the Scrum team:

- The Product Owner: One person that is responsible for maximizing the value of the product and the work that is performed by the Development team. S/he is the only responsible for managing the product backlog (an *artifact* explained after in this section).
- The Development team: Group of people (ideally between three and nine people) responsible for delivering the product increment at the end of each sprint. The team is self-organized, cross-functional and there are no sub-teams inside of it. They work according to the Product Owner decisions.
- The Scrum Master: One person that ensures that Scrum is well understood and that it adheres to theory, practices and rules. S/he is in charge of communication between the Scrum team and the people from the outside by helping useful interactions among them. S/he ensures that meetings are held and kept within the time-box.

A set of time-boxed events, called Scrum Events, is proposed in order to enable transparency and inspection during the project. Five events are defined:

- The Sprint: It is the heart of Scrum. During a time-box of one month or less, the team creates releasable product Increment. Each Sprint is considered as a project.
- Sprint planning: It plans the work to be done at the Sprint. It is created by all of the Scrum team. Two questions should be answered: What can be delivered at the end of the next Sprint? And, how will the work be achieved? A Sprint Goal, the objective set for the Sprint, is defined during this meeting.
- Daily Scrum: It is a fifteen-minute meeting to synchronize activities and create a plan for the day. It is conducted by the Development team and everyone should answer three questions regarding the Sprint Goal: What was done yesterday? What will be done today? And, are there any impediments to achieve it?
- Sprint Review: It is performed at the end of each Sprint in order to inspect the Increment and to adapt the Product Backlog. For this purpose, the Scrum team and the stakeholders collaborate about what was done in the Sprint.
- Sprint Retrospective: After the Sprint Review, and before the next Sprint planning, the Scrum team inspects itself and create improvements for the new Sprint.

Finally, Scrum Artifacts are defined in order to ensure transparency of key information allowing all the people involved in the project to have the same understanding. Three artifacts are defined:

- Product Backlog: It is an ordered list of everything needed in the product and represents the sole source of requirements for changes to the product. It is never complete and evolves along the project.
- Sprint Backlog: It is the set of Product Backlog selected for a Sprint and the plan to achieve both the Sprint Goal and the delivery of the Increment. It is a forecast by the Scrum team about functionalities to be done in the next Increment and the work needed to achieve them.
- Increment: All the Product Backlog items completed during a Sprint plus the value of the Increments of previous Sprints.

Agile software development methods and principles were introduced in this section. Interesting concepts regarding the principles behind agile methods such as flexibility, speed, leanness, learning, and responsiveness were presented. Two agile methods were introduced, Scrum and Extreme Programming. Such methods propose interesting agile approaches, however they are software-development oriented and some specific characteristics are out of the scope of this research. Other characteristics need to be adapted to agile processes, for instance, the “short iterations” principle used in agile software development methods, can be adjusted in order to define short activities of the process to increase its flexibility.

The second main domain where agility has been developed is *organisations*. The principles are presented in the following section.

2.2.2 Agile organisations

Agility in organisations is a widely studied domain in literature. In this study, the approach chosen to describe the constituent concepts of an agile organisation is the following: First, agile manufacturing, one of the most discussed subjects in the bibliography regarding agility, is presented. Second, an agility approach regarding the complete enterprise is detailed. Finally, agility in supply chains, the agile interaction between agile partners, is studied.

Agile manufacturing

Since 1980, manufacturing companies are facing unprecedented levels of globalization, socio-political changes and market instability. Several studies were led in order to provide clarification of the causes of these new conditions in business (Sharifi and Zhang, 1999) including a study by a group of scholars of the Iacocca Institute of Lehigh University in 1991 (Nagel and Dove, 1991). The report that resulted from the study, entitled “21st-century manufacturing enterprise strategy”, introduced for the first time the concept of Agile Manufacturing.

Even if several authors have defined agile manufacturing in their studies, there is no agreement on a unique definition. However, most authors agree that the main force, driver or dimension of agility is *change* (Vokurka and Fliedner, 1998; Sharifi and Zhang, 1999; Yusuf et al., 1999).

According to the study led at the Iacocca Institute of Lehigh University, agile manufacturing (Nagel and Dove, 1991) is *"…a manufacturing system with extraordinary capabilities (Internal capabilities: hard and soft technologies, human resources, educated management, information) to meet the rapidly changing needs of the marketplace (speed, flexibility, customers, competitors, suppliers, infrastructure, responsiveness). A system that shifts quickly (speed and responsiveness) among product models or between production lines (flexibility), ideally in real-time response to customer demand (customer needs and wants)"*. In (Sharifi and Zhang, 1999) agility is defined as "*the ability to cope with unexpected changes, to survive unprecedented threats of business environment, and to take advantage of changes as opportunities*". In (Yusuf et al., 1999), the authors define agility as "*the successful exploration of competitive bases (speed, flexibility, innovation, proactivity, quality and profitability) through the integration of reconfigurable resources and best practices in a knowledge-rich environment to provide customer-driven products and services in a fast-changing market environment*". (Goldman, 1995) defines agility as "*dynamic, context-specific, aggressively change-embracing, and growth-oriented. It is not about improving efficiency, cutting costs, or battening down the business hatches to ride out fearsome competitive storms. It is about succeeding and about winning profits, market share, and customers in the very centre of competitive storms that many companies now fear*." "*Agility is characterized by cooperativeness and synergism (possibly resulting in virtual corporations), by a strategic vision that enables thriving in face of continuous and unpredictable change, by the responsive creation and delivery of customer-valued, high quality and mass customized goods/services, by nimble organisation structures of a knowledgeable and empowered workforce, and facilitated by an information infrastructure that link constituent partners in a unified electronic network*" (Sanchez and Nagi, 2001). Finally (Katayama and Bennett, 1999) claims that agile manufacturing relates to the interface between the company and the market through a set of abilities for meeting widely varied customer requirements in terms of price, specification, quality, quantity and delivery.

Agile manufacturing core concepts have been identified by (Yusuf et al., 1999) as:

- Core competence management: Identification, improvement and affectation of the available set of skills.
- Virtual enterprise: Companies collaborate at corporate and operational levels.
- Capability for reconfiguration: Enterprises can easily adapt or reconfigure their business to answer a specific purpose.
- Knowledge-driven enterprise: Knowledge is the main differentiator of successful

business. Companies should be composed of people with the right set of competencies and knowledge.

An approach that allows to achieve agility in organisations is proposed by (Sharifi and Zhang, 1999). The authors propose a set of agility drivers that refer to the changes in the business environment that guide the company to modifying its strategy to remain competitive. The categories of these changes have been identified as: changes in the market, changes in competition criteria, changes in customer requirements, changes in technology, and changes in social factors. Moreover, the authors define four major categories of agility capabilities to help an organisation to appropriately respond to changes in its environment:

- Responsiveness: "Ability to identify changes and respond fast to them, reactively or proactively, and recover from them."
- Competency: "Extensive set of abilities that provide productivity, efficiency, and effectiveness of activities towards the aims and goals of the company."
- Flexibility: "Ability to process different products and achieve different objectives with the same facilities."
- Quickness: "Ability to carry out tasks and operations in the shortest possible time".

Furthermore (Kidd, 1996) claims that some of the key words related to agile manufacturing are: fast, adaptable, robust, virtual corporations, reconfiguration, dynamic teaming and transformation of knowledge ("*explicitly transforming raw ideas into a range of capabilities which are embodied in both products and services*").

The agile manufacturing approach has been introduced along with its principal characteristics. To summarize, common characteristics that describe an agile manufacturing organisation are: its speed, flexibility and responsiveness to reconfigure its resources in order to adapt to fast and unexpected change. Moreover, agile organisations collaborate with their partners and are knowledge-driven.

Enterprise agility

Enterprises must be ready to answer constant changes in the marketplace. This need drove the emergence of agile manufacturing concepts. Moreover, agile behaviour should be demonstrated on the management front (Izza et al., 2008). Agility is "*the ability to thrive in an environment of continuous and unpredictable change*" (Dove, 1994).

In Figure 6, a structure to construct agility in enterprises is proposed by (Dove, 1994). It is based on eight change domains (in the left side of the cube), since *change* is the focal point of agility; four agile dimensions (in the centre of the cube), in which change capabilities are in balance for the enterprise; ten agile attributes (in the higher part of the cube), enterprise characteristics that allow them to be agile; and twelve key enterprise elements (in the lower part of the cube), important sub-modules into which an enterprise can be decomposed from an agile perspective. Moreover, a focus is done on agile dimensions proposed by (Dove, 1994). If an enterprise wants to be agile, then it should have a good balance of change response transversal to the four dimensions of agility. The dimensions are: Cost and Time of change, and Robustness and Scope of the solution to face change.

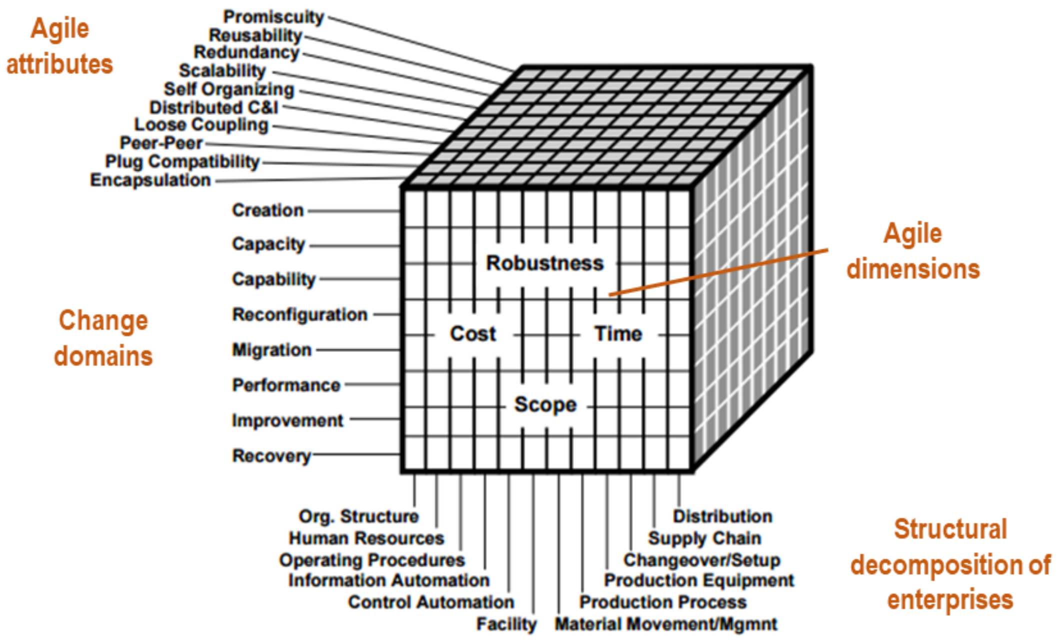


Figure 6: Structure of enterprise agility by (Dove, 1994)

A framework to measure agility is presented by (Izza et al., 2008). The five dimensions to measure agility of enterprise information systems are:

- Process dimension: Business processes can be measured in terms of time and cost to face changes.
- Organisation dimension: Organisational elements can be measured by their hierarchy, type, management type, among others.
- Information dimensions: Stored and manipulated information in the enterprise
- Resource dimension: people, IT and organisational structures
- Environment dimension: External factors to the enterprise.

The main concepts regarding enterprise agility have been presented. Many aspects of agility in manufacturing systems are retrieved in the domain of enterprise agility. As a global conclusion, change remains the driving force of agility.

Agile supply chains

Nowadays enterprises are aware that they need to be agile in order to survive and to be competitive. They also need to collaborate with suppliers and customers by working together to be truly competitive (Lin et al., 2006). A supply chain is "*a series of activities that moves goods from the customer order through the raw materials stage, supply, production and distribution of products and then to the customer*" (Ren et al., 2009). Supply Chain Management (SCM) states that companies should collaborate and integrate their business with their value chain partners in order to better respond to the customer needs (Agarwal et al., 2007). Business survival and success can be achieved through the definition of an *agile supply chain* (Ren et al., 2009). An agile supply chain, "*aims to enrich/satisfy customers and employees. It possesses the ability to respond appropriately to changes occurring in its business environment*" (Lin et al., 2006). Furthermore, it provides a new ground to innovate on the supply chain and allows to compete in the changing market (Balaji et al., 2014).

Agility in the supply chain is defined as "*the ability of an organisation to respond rapidly to changes in demand, both in terms of volume and variety*" by (Christopher, 2000). Agility "*describes a company or a supply chain that is able to change and adapt quickly to changing circumstances*", according to (Ren et al., 2009).

(Christopher, 2000) proposes four characteristics of agile supply chains:

- Market sensitive: Capability of reading and responding to real demand.
- Virtual: Agile supply chains use information technology to share data between buyers and suppliers.
- Process integration: Collaborative working between buyers and suppliers.
- Network based: Organisations that structure, coordinate and manage their relationships with partners.

Moreover, the principles behind these characteristics are discussed in the literature. For instance (Agarwal et al., 2007) define four characteristics of agile supply chain that follow the same principles presented by Christopher: Market sensitive and responsiveness, information-driven virtual integration, process integration and performance management, and centralized and collaborative planning.

Furthermore, a conceptual model of agile supply chain introduced by (Lin et al., 2006) is illustrated in Figure 7. Here, a compilation of agility drivers, capabilities, enablers and supply chain goals from literature is presented.

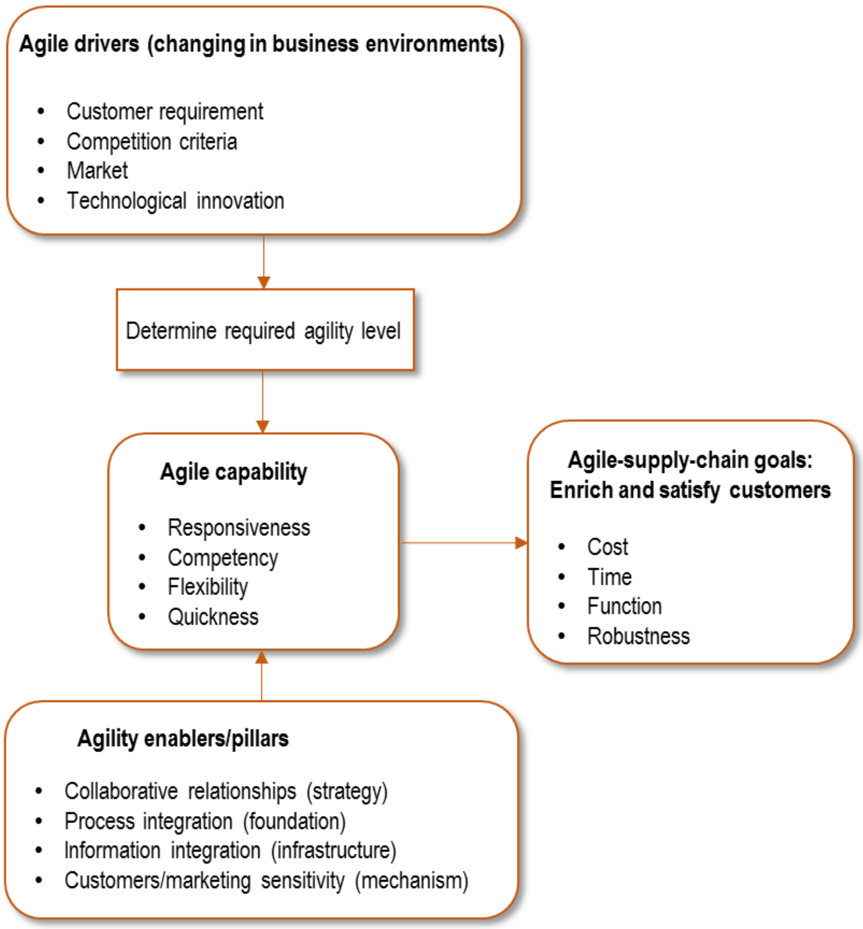

Agile drivers (changing in business environments)
Customer requirement
Competition criteria
Market
Technological innovation
Determine required agility level
Agile capability
Responsiveness
Competency
Flexibility
Quickness
Agile-supply-chain goals: Enrich and satisfy customers
Cost
Time
Function
Robustness
Agility enablers/pillars
Collaborative relationships (strategy)
Process integration (foundation)
Information integration (infrastructure)
Customers/marketing sensitivity (mechanism)

Figure 7: Conceptual model of agile supply chain by (Lin et al., 2006)

The concepts of agile manufacturing, enterprise agility and agile supply chain were presented. The main same principles and characteristics are retrieved in the three domains. The driving force of agility is *change*.

The third and last agility domains developed in this study are agile business processes and agile workflows. The principles are presented in the following section.

2.2.3 Agile business processes and agile workflows

Business processes were defined in section 1.1 as activities that a company engages to reach a business objective. Furthermore, effectiveness of business processes provides a key competitive advantage to the firm (Ray et al., 2004). Then, current companies must be prepared to rapidly adapt and adjust their business processes to face threats and changes in the global marketplace (Seethamraju and Krishna Sundar, 2013). This is how the concept of agility arises in the domain of business processes, in order to provide the continuous readiness of the system to detect and adapt to changes. Moreover, to achieve this agility, companies need to effectively identify and develop their set of capabilities in order to create new value propositions (Battistella et al., 2017).

Business Process Agility (BPA) is defined as "*the ability to add and/or reconfigure a business process by quickly adding new capabilities to the set of business processes capabilities in order to accommodate the potential needs of the firm*" (Raschke, 2010). The key concepts of BPA are flexibility and speed (Seethamraju and Seethamraju, 2009).

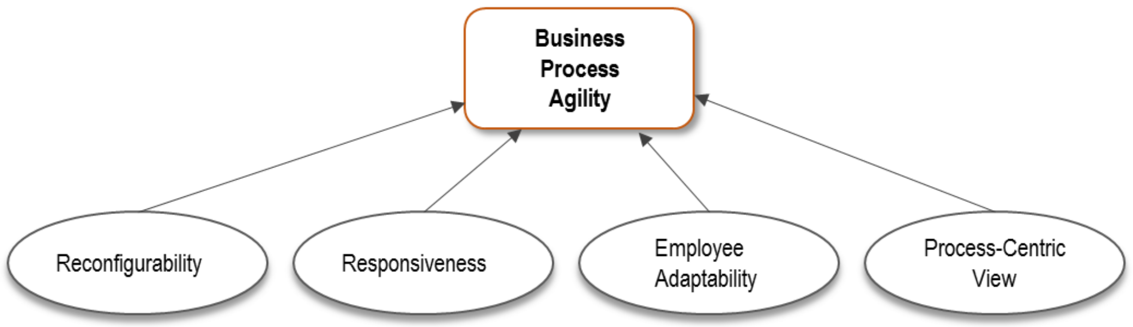


Figure 8: Business process agility construct by (Raschke, 2010)

Four components are identified as the core values of BPA by (Raschke, 2010), illustrated in Figure 8.

- Reconfigurability: Refers to the ability of adapting the process to changes in the environment.
- Responsiveness: Stands for the ability of identifying and recovering from changes (Sharifi and Zhang, 1999).
- Employee adaptability: Indicates the ability for people to adapt to changes.
- Process-centric view: It refers to the perspective of management of the process, contrary to the functional view.

Research in the domain of business process agility is focused on determining the relationship between Information Technology (IT) and agility in organisations (Seethamraju and Krishna Sundar, 2013). Firms must build process capabilities to improve IT, and consequently enable agility allowing them to achieve a competitive advantage (Sambamurthy et al., 2003). Moreover, digitized platforms of business processes such as enterprise resource planning (ERP), allow to create BPA because they help to identify changes and rapidly adapt business processes to match market requirements (Seethamraju and Seethamraju, 2009). ERP tools are integrated software systems composed of a set of modules that supports business processes (Robey et al., 2002).

In (Seethamraju and Krishna Sundar, 2013), the impact of ERP systems on BPA is studied through four characteristics.

- Process integration: It refers to the unification of activities, processes and information across units and functions (Seddon et al., 2010). The result of the study shows that integration helps to ensure efficient process changes, but a high dependence of information created by the system may turn on process changes low efficiency.

- Process optimization: It refers to the improvement (Seddon et al., 2010) and standardization of business processes. Standardization in enterprise systems regards the reduction of variability and variety of processes, information, business rules and technology systems. The result of the study shows that process agility is a requirement for customer-facing processes and not necessarily for transactional routine ones.
- Standardization of technologies: A standardized technology platform has a mixed effect on BPA. It depends on the nature of the business process (repetitive or not), on the degree of standardization and on when the simplification and improvement of the process took place (before or at the time of ERP implementation). Then, standardization may improve the ability to construct an agile process but in a limited way, since it can also constraint quickness to change the process.
- Best practices: This characteristic refers to the possibility of changing enterprise process to match the best practices contained in the ERP system. The result of this study shows that, while best practices improve efficiency, they do not necessarily contribute to agility. For the purpose of our study, this characteristic is taken into account in the knowledge and experience bases, where users can find past experiences to reuse in the current process (see section 3.5).

Furthermore, agility has been studied in the domain of Business Process Management (BPM). A proposition of agile BPM is detailed in (von Rosing et al., 2015). The authors apply principles and values of agile software development methods to the BPM philosophy through a way of working defined by BPM phases and elements:

- Agile analysis: Refers to the identification of the requirements of the process through collaborative and iterative work with the stakeholders. Requirements are prioritized and organized in short releases.
- Agile planning: A high-level project planning is defined including project releases, resources, cost and risks. Planning is also performed at other project levels such as release, iteration and daily.
- Agile architecture and design: A global process design is made from the to-be situation. Afterwards, through iterations small transition states incorporate detailed design.
- Agile build: Process architecture concepts are required in this phase. At first, the global architecture of the process is defined and it is detailed as the process evolves in short iterations.
- Agile testing: Testing is performed iteratively for each design object, then associated

requirements may be traced in the case of issues.

Finally, the last domain of agility detailed in this study is workflow agility. Workflow management systems "*are frequently used to control the execution of business processes and to improve their efficiency and productivity*" (Weber and Wild, 2004). The missing flexibility of workflow concept is a major limitation of traditional workflow management systems (Minor et al., 2014). The increasing demand for flexibility results in agile workflows (Bergmann and Gil, 2014).

In (Minor et al., 2014) the authors propose a case-based adaptation of workflows. They state that "*agile workflow technology addresses structural changes on workflow instances at run time*". Changes can refer, first, to *ad hoc* changes that can occur unexpectedly at any moment. Second, late modelling changes refer to structural changes that can be projected to some extent. The authors employ Case-based Reasoning (CBR) (see section 2.3.2) in order to adapt workflow instances. Past instances are recorded into *workflow adaptation cases*, and their changes are then adapted to new cases in a similar situation of change.

Likewise, in (Weber and Wild, 2004) the authors propose a theoretical approach to manage agile workflows, also based on CBR principles. The agile approach is based on the strict separation of the build-time and the run-time. During build-time, the preliminary model of the business process is created, focusing on business process aspects that have clear benefits or are critical parts of the process. During the execution of the workflow (run-time), if there are exceptions or changing requirements, changes are annotated in the model with context-specific information. CBR allows flexibility that enhances to adapt the workflow to respond to changing requirements.

Furthermore (Bergmann and Gil, 2014) describe a mechanism to retrieve similar workflows based on graph comparison based on process-oriented case-based reasoning (POCBR). POCBR "*aims at addressing the problem of creating new workflows as an experience-based activity*". The proposed approach consists of a general framework to represent workflows as semantically labelled graphs and to model related workflows knowledge intensive similarity measures including algorithms for similarity computations and retrieval.

Agile business process principles were presented. The concepts introduced in this domain, such as reconfigurability, flexibility and responsiveness, match previously presented concepts from agile software development and agile enterprise domains. Moreover, within the agile workflow domain, authors propose workflow adaptation using CBR principles, which affirms the necessity of knowledge-driven systems in agile processes. Nevertheless, in both domains some specific concepts, such as their link with information systems, are not applicable to our study.

Considering the above cited concepts and definitions of agility, a study of its features is proposed in the next section.

2.2.4 Synthesis of agile key characteristics

In order to define the drivers that make a process agile, the agile characteristics defined in the literature were taken into consideration, in particular from the above described domains of agile software development methods and agile manufacturing, in order to reuse and adapt them and define our own agile process. At a first time, all concepts and characteristics from literature related to the concept *agility* were outlined, as illustrated in Figure 9. Some concepts from other domains were identified as key agile characteristics by the research team (e.g. concurrent engineering).

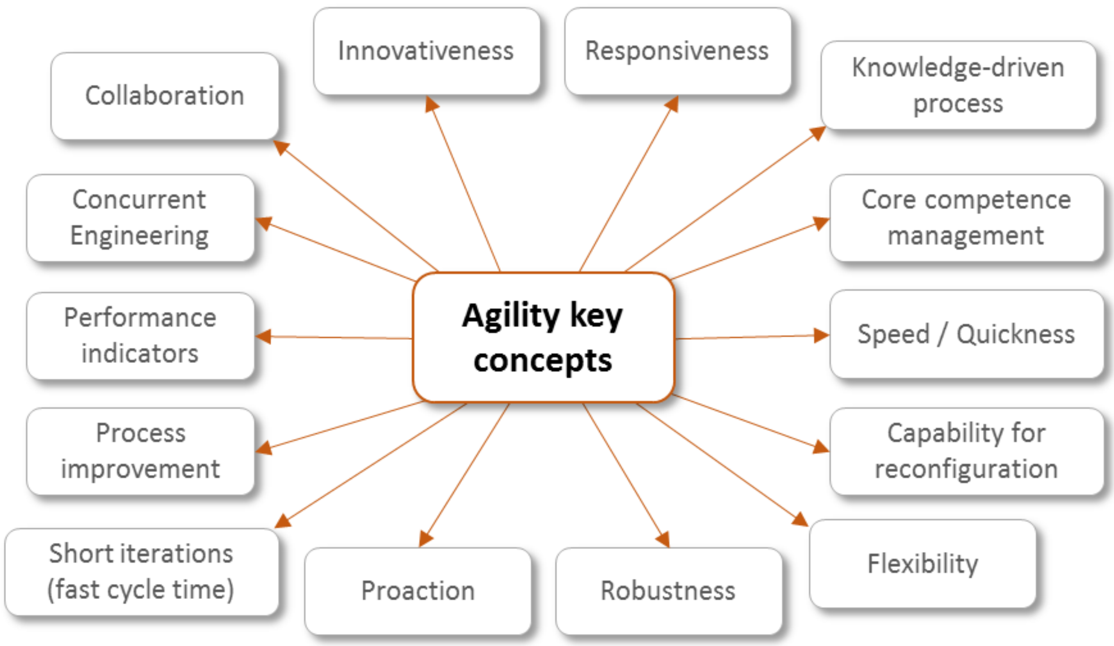


Figure 9: Agility key concepts and characteristics

Based on characteristics described in the last sections and, mainly, on concepts and characteristics identified in Figure 9, the key concepts of an agile process were defined. For this purpose, brainstorming sessions were carried out among the academic and industrial partners of this research work. The ten key concepts (values, characteristics and requirements) of agile processes that have been identified are:

- Capability for reconfiguration: Ability to easily and significantly change activities of an agile process to answer new purposes, constraints or events,
- Collaboration: Association with team members and with other enterprises or individuals in order to solve a problem or make a decision,
- Concurrent Engineering: Integrated organisation path for agile processes in which the activities overlap and all the departments collaborate from the beginning of the process (Valle and Vázquez-Bustelo, 2009),
- Core Competences Management: Knowledge of the available set of skills, its continuous improvement and its affectation to the adequate work position,
- Innovativeness: Continuous engagement to search and experiment new ideas,
- Knowledge-driven Process: Ability to reuse knowledge and experiences through the process,
- Proaction: Actions taken to predict and adapt to change before it occurs,
- Responsiveness: Ability to identify changes (expected and unexpected), respond fast, reactively or proactively, and recover from them (Sharifi and Zhang, 1999),
- Robustness: Ability to tolerate all transitions caused by change without having to take corrective actions (Conboy and Fitzgerald, 2004),
- Short activities: Splitting of long tasks to increase flexibility.

From this list of ten key concepts, the requirements about our global agile knowledge and experience based process model can be defined. An agile process needs to have capabilities for reconfiguration, responsiveness, robustness, proaction, innovativeness. It should be structured and organized taking into account concurrent engineering, collaboration, core competence management and short activities perspectives and principles. A special focus is done on the concept of knowledge-driven process, considered as a major driver of agile processes in our work. An agile process needs to continuously be guided by past experiences and knowledge in order to avoid undesirable past situations and to promote successful ones. However, it is important to notice that some of the concepts are not treated in this study such that collaboration concept, which we consider out of the scope of this work. Nevertheless, it may be studied as an extension of this work.

In this first section, existing works regarding agility concepts were presented. The values and principles of agile software development methods were outlined including an overview of two methods: Scrum and Extreme Programming. The second agility domain discussed was agile organisations. For this purpose, agility in manufacturing companies was detailed followed by a global enterprise approach and agility through the supply chain. Agile business processes and agile workflows were defined along with its principles and concepts. Finally, a summary of agility characteristics was presented followed by the introduction of the selected key concepts which define the first global requirements of our agile process. Considering the need for a knowledge-driven process, the principles of knowledge based systems are described in the next section.

2.3 Knowledge and experience capitalization and reuse

Knowledge and experiences capitalization and reuse are well-known practices that support problem-solving processes and, thus, contribute to enterprise performance. Moreover, nowadays the competitive advantage of an organisation resides on the "*know-how* (practical knowledge) and the *know-what* (formal or cognitive knowledge)" of the people who create and exploit knowledge in companies (Zorn and Taylor, 2004). In a company, then, best practices and experiences from a specific situation should be properly stored in order to allow their future reuse, in a way to avoid problem-solving from scratch.

Knowledge-based systems (KBS) are a system architecture used to support human problem solving. Their benefits can be summarized as: faster decision making, increase of productivity, increase of the quality of decision making (Schreiber and Akkermans, 2000). They help to solve complex problems through the use of human knowledge captured in an information system (Turban et al., 2004). Standard KBS methods exist such as KADS (Knowledge Acquisition and Design Structuring) and CommonKADS (Schreiber and Akkermans, 2000), which seeks to capture the knowledge used by people to do a given task and transform it into a system implementation (Kamsu Foguem et al., 2008). Moreover, Knowledge Management (KM) (Liebowitz, 2001; Zorn and Taylor, 2004; Dalkir, 2013) is "the process of creating value from an organisation's intangible assets" (Liebowitz, 2001). It refers to the management system that allows to define, develop, control and exploit a company's expertise (Zorn and Taylor, 2004). The knowledge management approach seeks to identify and capture all intellectual assets in order to transfer and reuse them across the company, with the purpose of creating value-added benefits to the organisation (Liebowitz, 2001). Knowledge management systems capitalize the generic knowledge necessary to solve a problem through models of expertise, creating a high level of abstraction of models that makes them difficult to be adopted (Kamsu Foguem et al., 2008). For this reason, our work is based on experience feedback principles over knowledge management ones.

In this section, first the notions of knowledge and experience are stated. The principles behind the experience feedback approach are detailed next. Finally, the link between knowledge and experiences capitalization and reuse and agility is explained.

2.3.1 The notions of knowledge and experience

There are different manners to describe and to refer to the concepts of knowledge and experience in the literature. Before explaining the core principles of knowledge and experiences capitalization and reuse, these concepts need to be clarified.

In order to characterize the term of knowledge, several approaches are found on literature. One of the most spread computer-science-perspective refers to the triplet: Data, information and knowledge (Bergmann, 2002). Data are "*syntactic entities (i.e. data are patterns with no meaning). Data can be stored and processed by computers.*" Information is "*interpreted data (i.e. data with meaning). Hence, information is data together with semantics.*" Knowledge is "*a set of related information with pragmatics. Knowledge puts information into a context given by certain task or goal*." From this classical perspective and its definitions proposed by Bergmann, an adaptation is proposed in (Béler, 2008). The author adds to the known data/information/knowledge triplet, the concept of experience, as illustrated in Figure 10. The gradual transformation (or hierarchy) information, experience, knowledge is explained in (Kamsu Foguem et al., 2008): information corresponds to an event along with its context; an experience permits to formalize analysis and solution. Finally, when lessons learned, procedures, rules, etc., are implied from past experiences, knowledge is obtained. In order to clearly differentiate concepts, *knowledge* and *experience* are further described below.

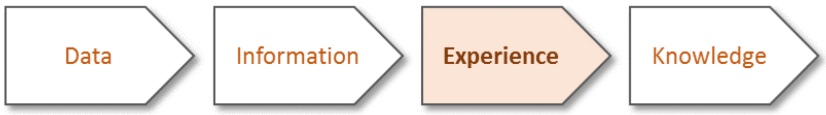


Figure 10: Positioning of an experience in the triplet Data-Information-Knowledge. Adapted from (Béler, 2008)

According to (Bergmann, 2002), an experience is "*valuable, stored, specific knowledge that was acquired by a problem solving agent in a problem solving situation*". Thus, an experience is a piece of contextualized knowledge obtained from a previous situation that can be capitalized in order to be reused in the future. In our study, and as detailed in section 3.5, every piece of experience is stored in a dedicated repository denominated "Experience Base" (EB).

Knowledge, on the other hand, has a higher level of generalization and a larger scope such as general rules, constraints, mathematical laws, etc. (Dalkir 2013). Knowledge has been either defined or validated by knowledge experts. "*Knowledge corresponds to high added-value information that allow to generate, to infer new information*" (Béler, 2008). In our study, and as detailed in section 3.5, every piece of knowledge is stored in a dedicated repository denominated "Knowledge Base" (KB).

Moreover, experiences can be transformed into new knowledge. In her work (Rakoto, 2004) explains that knowledge such as rules or company referential may be generated from a set of experiences. This is possible through the generalization of experiences to create knowledge with the intervention of experts who ensure that knowledge will help to avoid future negative events and to foster future positive events. It is important to notice that the process of experience generalization into knowledge is considered but not detailed in this study.

Experience feedback principles are detailed in the next section.

2.3.2 Experience feedback principles

Organisations performance can be improved through the sharing of experiences and lessons learned. The capitalization and reuse of knowledge and/or experiences in organisations is a widely discussed subject in the literature through different approaches. Knowledge management principles were introduced at the beginning of section 2.3. However, it has been stated that this research is focused on experience feedback principles.

The objective of experience feedback (EF) approach is to define a manner to capitalize past experiences and lessons learned in order to reuse them in future similar situations. Experience feedback can be considered as an organisational process seeking to grow the knowledge bases on an organisation through the experience of its actors (Béler and Desforges, 2007). Moreover, experience feedback can be defined as "a structured method of capitalization and exploitation of information resulting from the analysis of negative and/or positive events. It implements a set of human and technological resources which must be managed to contribute to reducing the repetitions of errors and to support some best practices" (Rakoto, 2004). The experience feedback process helps to avoid or to foster the reuse of respectively, negative (failures, problems) or positive (innovation, improvement) past experiences (Romero Bejarano, 2013). According to (Kamsu Foguem et al., 2008), the major experience feedback advantage is that it allows knowledge contextualization since a particular problem resolution process is capitalized and can be partially or totally reused. Moreover, as previously cited (Kamsu Foguem et al., 2008) argues that experience feedback approach gradually builds knowledge from useful cases, through the mentioned data/information/experience/knowledge hierarchy. Close to experience feedback, experience management can be considered a special form of knowledge management restricted to managing experiences (Bergmann 2002). According to (Bergmann, 2002), the experience management process is defined through the steps of experiences collection, modelling, storage, reuse, evaluation and maintenance.

Literature reviews other techniques allowing to reuse knowledge contained in past experiences, especially the Case-based Reasoning (CBR) model (Kolodner, 1993; Aamodt and Plaza, 1994) which proposes to reuse previous experiences to solve current problems. It differs from experience feedback and knowledge management principles with the sense that it is regarded as a cognitive model and a technical architecture instead of an organisational model for experience reuse (Bergmann, 2002). CBR principles are used in this work as a baseline to define an agile process lifecycle (see chapter 4). CBR follows a cycle, illustrated in Figure 11, composed of four main phases (Aamodt and Plaza 1994):

- A retrieval phase is performed from previous cases, and the most similar case (retrieved case) to the problem (new case) is found.
- The information or solution contained in the retrieved case is reused (adapted) to solve the current problem, and a solved case is suggested.

- The suggested solution is verified and revised by the user, confirming the solution.
- The repaired case is retained in the case base for future problem solving.

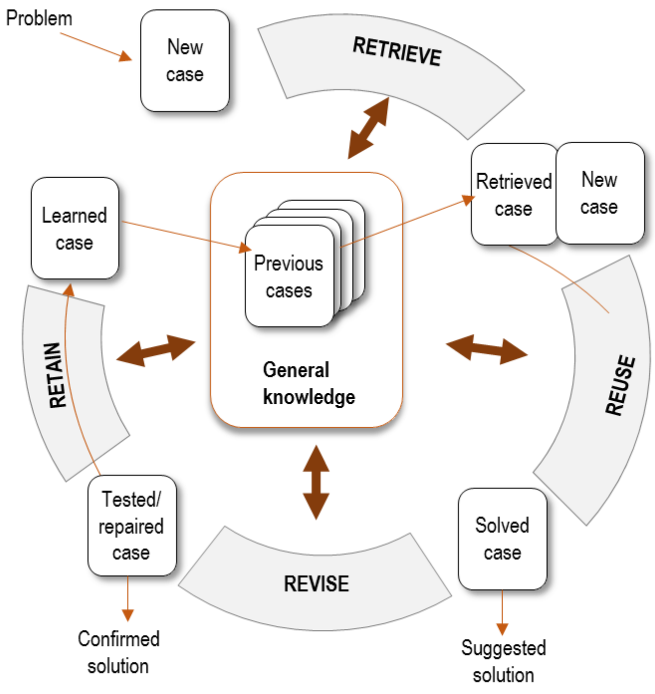


Figure 11: The CBR cycle by (Aamodt and Plaza, 1994)

2.3.3 Link between agility and knowledge and experience capitalization and reuse

In section 2.2.4 key agile characteristics and concepts were listed. Moreover, as it was mentioned, the concept of “knowledge-driven process” is considered as a major driver for agile processes. According to (Qumer and Henderson-Sellers, 2008) “learning” is one of the five features of agile software development methods, it refers to the continuous application of updated knowledge and experiences. On the other hand, in the domain of agile enterprises and agile workflows, the knowledge capitalization and reuse is of major importance. In (Yusuf et al., 1999) the authors refer to knowledge-driven enterprise as the ability to use the collective company knowledge and skills of the people for competitive product creation. In the domain of agile workflows, CBR is used as a base to define a model for knowledge capitalization and reuse (Bergmann and Gil, 2014; Weber and Wild, 2004). Even if knowledge-driven process is a key characteristic of different agile methods, to our knowledge, there is no agile approach that defines and implements complete knowledge and experience capitalization and reuse mechanisms.

An agile process needs to be guided by experience and knowledge capitalization and reuse mechanism. Experience feedback principles are used in this work in order to define a structured approach to capitalize experiences and reuse them to support decision making in future similar situations and then contribute to process reconfiguration. Moreover, CBR principles are considered and adapted to our work in order to define the lifecycle of an agile process (see chapter 4).

In the second bibliography section, a study on knowledge and experience capitalization and reuse was presented. First, the concepts of *experience* and *knowledge* have been detailed. Experience feedback principles were described followed by case-based reasoning ones. Finally, the link between the concepts introduced in this section and agility concepts detailed in section 2.2 was presented. The third and last research pillar of this study, problem solving, is introduced in the next section.

2.4 Problem Solving

In companies, continuous improvement needs to be ensured by the management in order to gain efficiency to compete in the industrial environment. Problem solving is a key activity that enterprises perform on a daily basis to improve quality and to obtain sustainable and continuous improvement (Kamsu Foguem et al., 2008). It refers to the analysis of a problem in order to find its root cause(s), to implement a solution and to avoid its recurrence. Different problem-solving methods exist and are widely applied in industry, mostly in order to solve manufacturing problems in order to guarantee the quality of the products.

Industrial problem-solving processes are of major importance to this research since they concern the direct application of the agile knowledge and experience based process model. Section 2.4.1 discusses the problem-solving approach, and some methods used for industrial problem solving are described in section 2.4.2. The links between problem solving and the two above presented research domains (agility and knowledge and experience capitalization and reuse) are introduced in section 2.4.3.

2.4.1 Problem-solving approach

Problem solving is a widely discussed topic on literature since it regards different application domains. Problem solving includes the processing of information concerning a deviation from a desired state, to identify and choose the appropriate action to reduce the deviation from the planned situation (Newell and Simon, 1972).

Furthermore, problem-solving processes are widely discussed in literature for being a key characteristic of continuous improvement. Continuous improvement is a requirement of the ISO 9000 world standard which includes the improvement of products and services, the treatment of non-conformities and the improvement of the performance of the quality system (International Organisational for Standardization, 2015). According to this standard, a robust process should be in place to treat non-conformities and problems. More specifically, an industrial technical problem "*appears when the performance of a product or one of its components deviates from what has been stated on standards or specifications and there is a convergence of unknown factors and unusual or unexpected conditions that make complex for the firm to bring the product back to its expected performance*" (Romero Bejarano, 2013). Then, an industrial problem-solving process is "*a set of activities allowing to solve in an organized and systematic manner complex problems. This approach is usually based on rules, principles, expert knowledge and it can employ, in a structured and logical way, a set of technical tools*" (Jabrouni, 2012). For the purpose of this research, the latter definition of problem solving is considered.

A problem-solving process is deployed when a problem occurs. Then, the main goal of the problem-solving process is to find the root cause (s) of the problem, to treat it (them) and to avoid its (their) future recurrence. This problem-solving approach is valid for most problem-solving methods that are detailed in the next section.

2.4.2 Problem-solving methods

Taking into consideration problem-solving approaches described in literature to solve industrial problems, two major categories can be distinguished. The first one regards innovation and invention as the drivers of the methodology, such as the widely known “theory of inventive problem solving” (TRIZ in its Russian abbreviation) (Fey et al., 1994; Ilevbare et al., 2013; Czinki and Hentschel, 2016). This theory is based on theoretical conclusions and on the analysis of numerous patents from the past in order to propose an adapted solution to the current problem (Czinki and Hentschel, 2016). TRIZ is mostly used for innovation problems such as product design and business development (Ilevbare et al., 2013). This technique is excluded from our research given its inventive nature and the type of problems it helps to solve.

The second category of problem-solving methods refers to several structured methods and approaches that exist to solve problems in the domain of quality improvement (Choo et al., 2015). These are the methods that lie at the heart of this research. Several standard problem-solving methods exist to solve problems within companies. Some of the most common and known methods are: Plan-Do-Check-Act (PDCA), 8 Disciplines (8D), and six sigma DMAIC (Define, Measure, Analyse, Improve and Control) (Kamsu Foguem et al., 2008). They are detailed in the next sections.

PDCA (Plan-Do-Check-Act)

The PDCA (Plan-Do-Check-Act) cycle (also known as the Deming cycle, PDSA (Plan-Do-Study-Act), and also as the Shewhart cycle) (Shewhart, 1930; Deming, 2000), consists of four stages to follow in order to solve a problem (Prashar, 2017). The first step refers to the development of a Plan for a change in the process or system. The second step refers to the implementation of the plan. The third step refers to the study and the analysis of the data collected during the Do phase in order to verify its consistency with the expected results. The last phase refers to either adopting or abandoning the change according to the Check results. Then, the first step is reached again, either to restart with a new change, or to continuously improve the cycle, as illustrated in Figure 12.

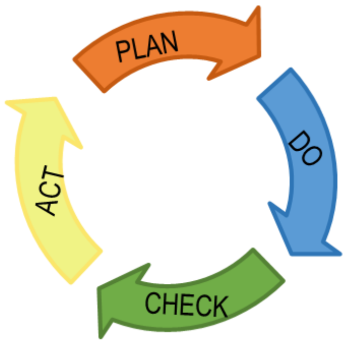
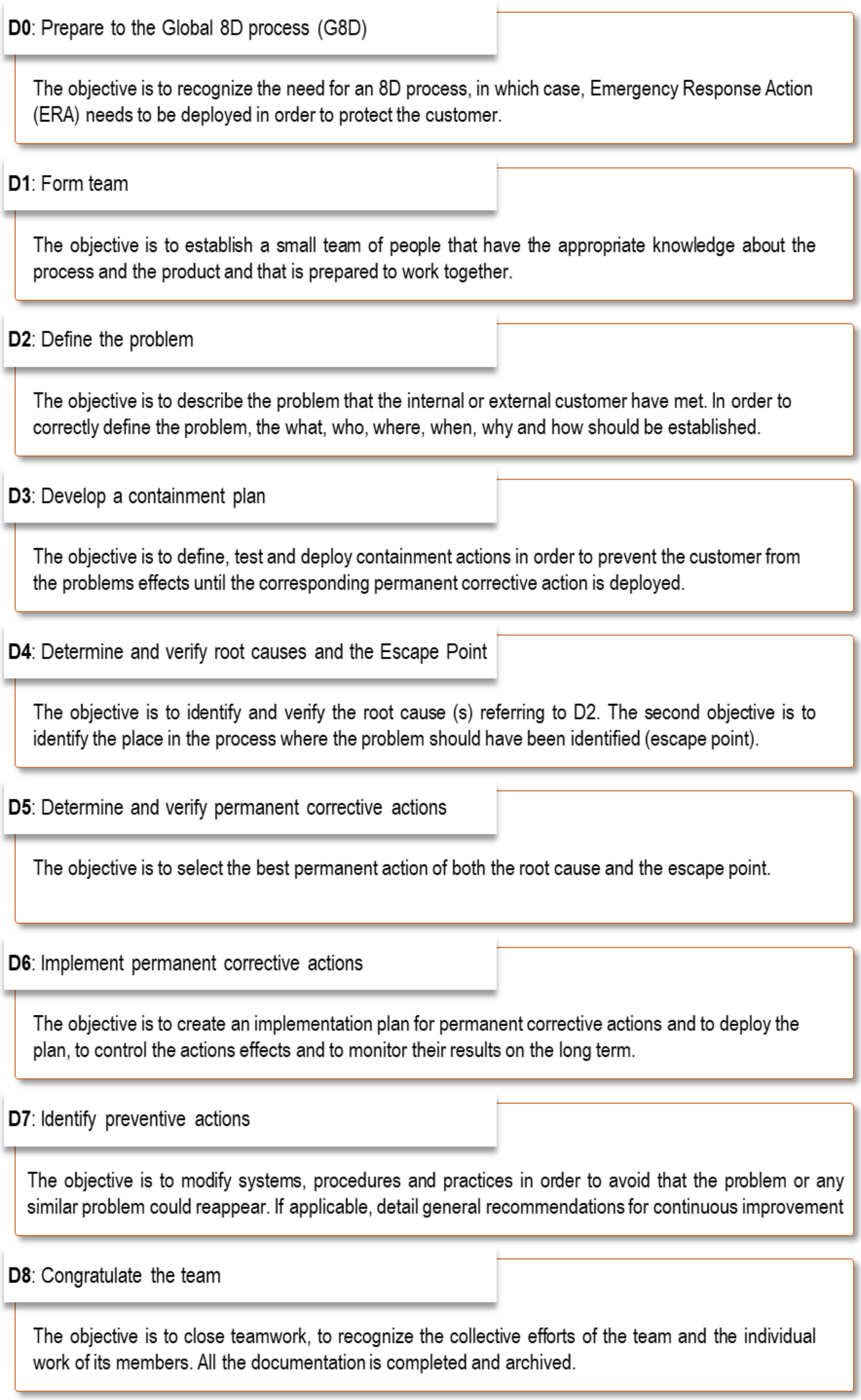


Figure 12: The PDCA cycle adapted from (Deming, 2000)

8D (8 Disciplines or 8 Do)

The 8 Disciplines (8D) is a problem-solving method developed and first used within the Ford Company in the late 1980s (Jabrouni, 2012). The methodology proposes eight structured steps in order to solve a problem based on team work (Riesenberger and Sousa, 2010). It is usually applied in companies as the standard approach to solve common problems (Duret and Pillet, 2011). The eight steps or disciplines are illustrated in Figure 13. Moreover, the 9-Steps (9S) (IAQG (International Aerospace Quality Group), 2014) method considers the 8D steps and includes a first step where immediate containment actions are carried out.



D0: Prepare to the Global 8D process (G8D)

The objective is to recognize the need for an 8D process, in which case, Emergency Response Action (ERA) needs to be deployed in order to protect the customer.

D1: Form team

The objective is to establish a small team of people that have the appropriate knowledge about the process and the product and that is prepared to work together.

D2: Define the problem

The objective is to describe the problem that the internal or external customer have met. In order to correctly define the problem, the what, who, where, when, why and how should be established.

D3: Develop a containment plan

The objective is to define, test and deploy containment actions in order to prevent the customer from the problems effects until the corresponding permanent corrective action is deployed.

D4: Determine and verify root causes and the Escape Point

The objective is to identify and verify the root cause (s) referring to D2. The second objective is to identify the place in the process where the problem should have been identified (escape point).

D5: Determine and verify permanent corrective actions

The objective is to select the best permanent action of both the root cause and the escape point.

D6: Implement permanent corrective actions

The objective is to create an implementation plan for permanent corrective actions and to deploy the plan, to control the actions effects and to monitor their results on the long term.

D7: Identify preventive actions

The objective is to modify systems, procedures and practices in order to avoid that the problem or any similar problem could reappear. If applicable, detail general recommendations for continuous improvement

D8: Congratulate the team

The objective is to close teamwork, to recognize the collective efforts of the team and the individual work of its members. All the documentation is completed and archived.

Figure 13: The Ford 8D method adapted from (Jabrouni, 2012)

DMAIC (Define, Measure, Analyse, Improve and Control)

DMAIC (standing for Define, Measure, Analyse, Improve and Control) is a structured procedure for improvement in the domain of quality management (de Mast and Lokkerbol, 2012). This problem solving technique is part of the six-sigma method. Six sigma is "*an organized and systematic method for strategic process improvement and new product and service development that relies on statistical methods and the scientific method to make dramatic reductions in customer defined defect rates*" (Linderman, 2003). Unlike the 8D method, based on a qualitative approach (expertise), DMAIC method is based on a quantitative approach (statistics). The five phases of the DMAIC described by (Boon Sin et al., 2015) are presented in Figure 14. Moreover, some authors add more steps to the DMAIC, for instance (Duret and Pillet, 2011) describe the Standardization step (dmaicS) where the solution is sustained, best practices are applied and the project is finished.

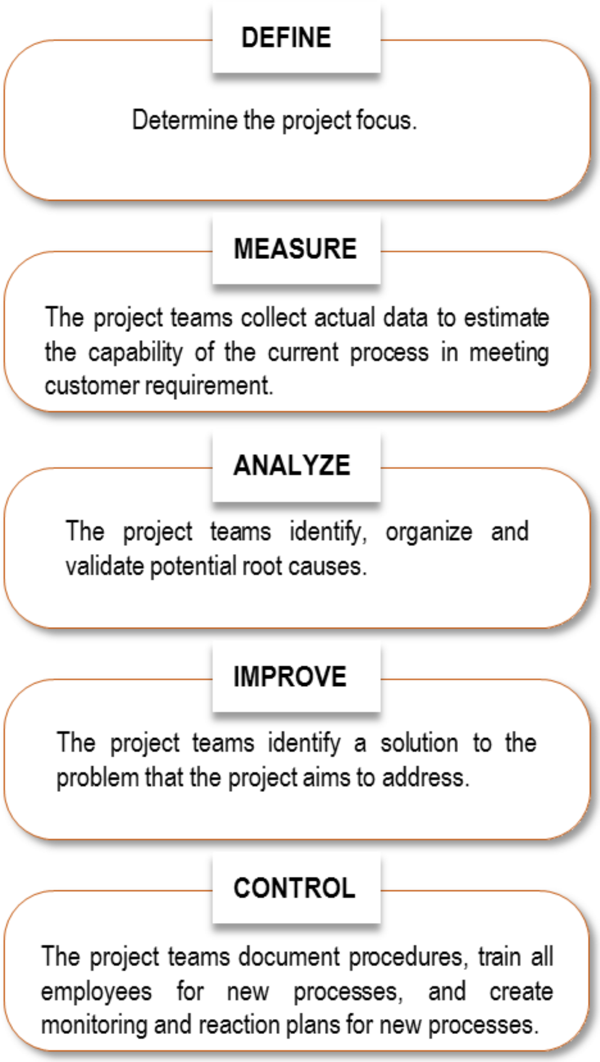

DEFINE
Determine the project focus.
MEASURE
The project teams collect actual data to estimate the capability of the current process in meeting customer requirement.
ANALYZE
The project teams identify, organize and validate potential root causes.
IMPROVE
The project teams identify a solution to the problem that the project aims to address.
CONTROL
The project teams document procedures, train all employees for new processes, and create monitoring and reaction plans for new processes.

Figure 14: DMAIC phases adapted from (Boon Sin et al., 2015)

Other methods to solve complex problems exist such as the Shainin system (Bhote, 1991) which consists of the use of a set of 24 statistical technics to solve a problem and improve quality. The Kepner Tregoe method (Kepner and Tregoe, 1982), also called Problem Solving and Decision Making (PSDM), proposes four processes to solve a problem: situation analysis, problem analysis, decision analysis and potential problem analysis.

Considering the previously presented structured problem-solving approaches (PDCA, 8D, DMAIC), common phases are identified among those methodologies. Four standard phases can be distinguished: Context (actions conducted to fully understand the problem and its context), Analysis (deep analysis to find the root causes of the problem), Solution (definition of the actions, application and verification of their effectiveness) and Generalization/Lesson learned (standardization of solutions and, if possible, their generalization for potential reuse in future similar situations) (Béler, 2008; Kamsu Foguem et al., 2008; Jabrouni, 2012; Romero Bejarano, 2013).

2.4.3 Links between problem solving, agility, and knowledge/experience capitalization and reuse

The purpose of this section is to detail the intersection between the research domains: agility, knowledge and experience capitalization and reuse, and problem solving.

The problem-solving methods considered and studied in this work (PDCA, 8D, and DMAIC) are known to be sequential and structured processes. A lack of flexibility to dynamically adjust the problem-solving process to fit with simple to complex, dynamic and networked contexts and problems has been observed when these methods are applied in industry. More specifically, and as it has been mentioned in the introduction of this research work, problem-solving processes present a dual structured/exploratory nature. In order to define whether agility concepts are included in problem-solving processes or not, the exploratory and structured nature of the process are considered separately.

First, regarding the exploratory activities or steps, some agility key characteristics arise such as a high level of flexibility that allows the reconfiguration of the process through different options. When an unexpected situation appears, the process can be adjusted to achieve the objectives. Nevertheless, a low level of formalization is denoted by the high level of flexibility, since the determination of standards for their systematic reuse is hard to achieve. Therefore, the capitalization and reuse of knowledge and experiences is difficult to accomplish and it can be said that the level of agility in such a process is “low”. On the other hand, the structured part of the process is represented as a set of pre-defined activities, which allow their systematic reuse. The advantages of such a process are, first, that the activities of the process can be carried out without uncertainty and, second, that decision makers can be supported by the formalization and reuse of knowledge. However, it is quite difficult to adapt the process to unexpected events. Moreover, the capitalization and reuse of experiences only concern activities because decision making is already standardized through knowledge. Hence, there is no agility in the process to react to problems.

Therefore, an agile problem-solving process needs to be defined in order to provide flexibility and adaptability to the existing processes through the necessary combination of structured and exploratory process components. To our knowledge, this topic has not been yet treated on literature and it constitutes a major challenge to our study.

On the other hand, a link between problem solving and knowledge and experiences capitalization and reuse exists and is discussed in literature. First, most problem-solving methods (especially the ones treated within the scope of this study) include a step of "standardization" or "transfer of knowledge" with the purpose of capturing and consolidating the learning in order to enhance its generalization (Romero Bejarano, 2013). Furthermore, the standardization step seeks to spread best practices into other workplaces (Duret and Pillet, 2011). Then, a process can be built on the basis of a similar process that was performed in the past, adapting it to the current context. Moreover (Kamsu Foguem et al., 2008; Jabrouni et al., 2011) describe how methods for capitalizing and reusing past experiences can improve problem-solving processes and consequently, the performance of organisations over time. However, a disadvantage of the current problem-solving methods is that even if many modern enterprises ensure that they perform capitalization and reuse of experiences, most of the time the "standard capitalization/reuse cycle" is not correctly performed or it is not adequate (e.g. the resolution of the problem is capitalized into a spreadsheet such as Excel, but its reuse is not performed). This can be seen as a problem of lack of capitalization and reusing tools or software in today's organisations. Then, the simplification of experiences capitalization phase, in order to reinforce their reuse, is one of this research's objectives.

Problem-solving concepts and methods were presented in this section. Three of the methods presented in this section are taken into consideration further in this work: PDCA, 8D, DMAIC. Such processes are considered in our study for having a dual nature, they are both structured and exploratory process. Then, problem-solving processes could be improved by agile principles.

2.5 Synthesis and Contributions

The three domains of research of our work were presented in this chapter. First; agility domain was introduced detailing its different application areas, its principles and finally its key concepts and characteristics. Second, knowledge and experience capitalization and reuse concepts, experience feedback and CBR principles were discussed. Finally, industrial problem-solving concepts were described including some of the commonly used methods by industrial companies.

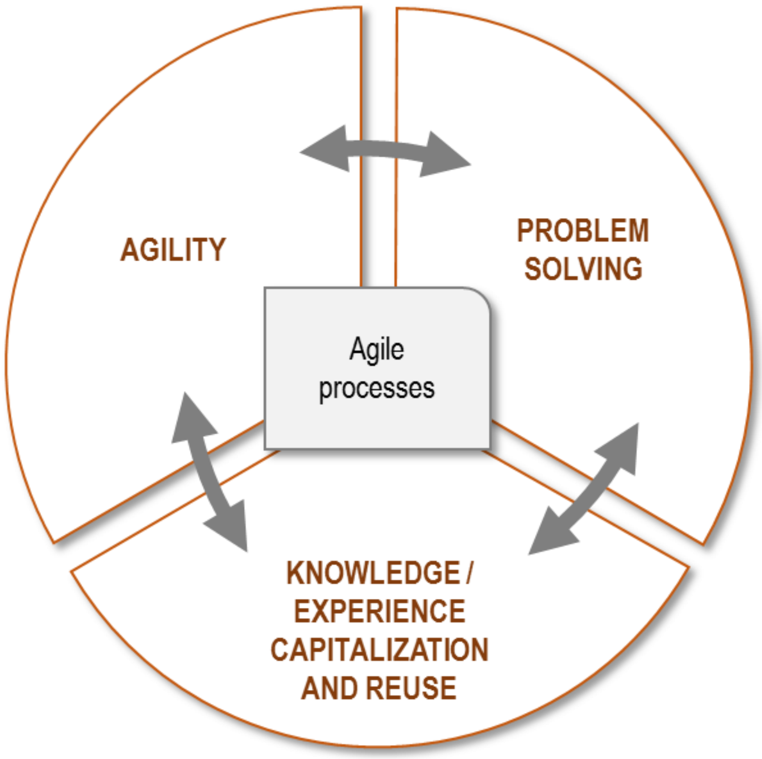


Figure 15: Interactions between research domains

The links between the three research domains were presented. The interactions between them are illustrated in Figure 15. A problem-solving process allows the structuration of an experience feedback approach, which supports the company knowledge management. In order to work better, problem solving can exploit formalized knowledge and to use knowledge-based systems tools such as CBR. Furthermore, problem-solving processes need to be agile in order to face changes in their environment and to adapt quickly. Then, agility principles such as responsiveness and flexibility can help to improve problem-solving processes. Regarding agility, agile processes need to be knowledge-driven in order to learn from past experiences. Then, knowledge-based systems such as experience feedback allow supporting the capitalization and reuse of agile experiences.

To our knowledge, there is no specific method combining problem solving, agile methods and a unified and complete approach including experience and knowledge capitalization and reuse principles.

The contribution of this thesis regards the proposition of an *agile problem-solving process based on the capitalization and reuse of experiences and knowledge.* The model defined in this work is composed of an information model for agility and of an agile process lifecycle, based on CBR principles, which allows to define and to execute an agile process.

The objectives of this research were described in section 1.3. Considering all concepts and principles introduced in this state of the art chapter, such objectives are further detailed below.

- The first objective of this research work regards the adaptation of agile principles in order to improve problem-solving processes. From all the agile domains, ten key concepts were identified which need to be applied to problem-solving processes in order to make them more agile. In particular, existing problem-solving processes need to be supported with knowledge and experience capitalization and reuse concepts.
- The second objective considers that specific algorithms and mechanisms, based on experience feedback principles, could help decision makers to define and execute processes through the reuse of similar past cases. Then, decision making can be supported by a set of indicators based on past situations.

3. INFORMATION MODEL FOR AGILITY

3.1 Introduction

In the previous chapter, through the literature review, the research statement concerned by this study was defined. A process that can be defined through agility principles and driven by experiences and knowledge needs to be developed. For this purpose, this chapter aims at defining the information model for agility in order to clearly state its principles and elements.

The first part of this chapter outlines the definition of the key concepts of the proposed agile process. The second part of this chapter describes the knowledge and experience capitalization and reuse mechanism. Finally, an illustration is proposed in order to exemplify the introduced concepts.

3.2 Definition of the concepts of the agile process

As defined in section 1.1, a process is "*a specific ordering of work activities across time and place, with a beginning, an end, and clearly identified inputs and outputs*" (Davenport, 1993). Considering the agile key characteristics and knowledge/experiences capitalization and reuse principles described in chapter 2, the agile process can be defined. The definition of the agile process is presented followed by the explanation of its components.

Our agile knowledge and experience based process is composed of activities and decision-making points. It aims at satisfying constraints and ensuring objectives and process efficiency. Supported decision making enables to continuously adapt the process during its execution. Our agile process is based on knowledge and experiences capitalization and reuse principles. Figure 16 illustrates the agile process along with its elements: a sequence of activities and of decision-making points including decision support (dashboards) relying on knowledge and experience bases.

Activities and decision-making points are the building blocks of the agile process. An activity is an indivisible element defined by its identifier (ID), its description, and its cost and delay values. It means that for a given activity, its purpose and/or nature is identified through the description. The cost of the activity is denoted with its Cost value, and the time necessary to carry out the activity is identified with its Delay value. The activities that compose a given process are denoted "scenario activities". All scenario activities are linked to their corresponding "generic activity" which reflects the objective of the activity that will be respected for all scenario activities (same description). For instance, in a problem-solving process (see section 2.4) there is an activity intended to define a team to solve the problem. Then, the generic activity description should be "team definition", and all scenario activities should be named after it. An agile process is composed of an indeterminate number of activities since at the beginning, a first version of the process is defined with a certain number of ordered activities and, afterwards during process execution, as the process can change following the different decisions, activities may be added.

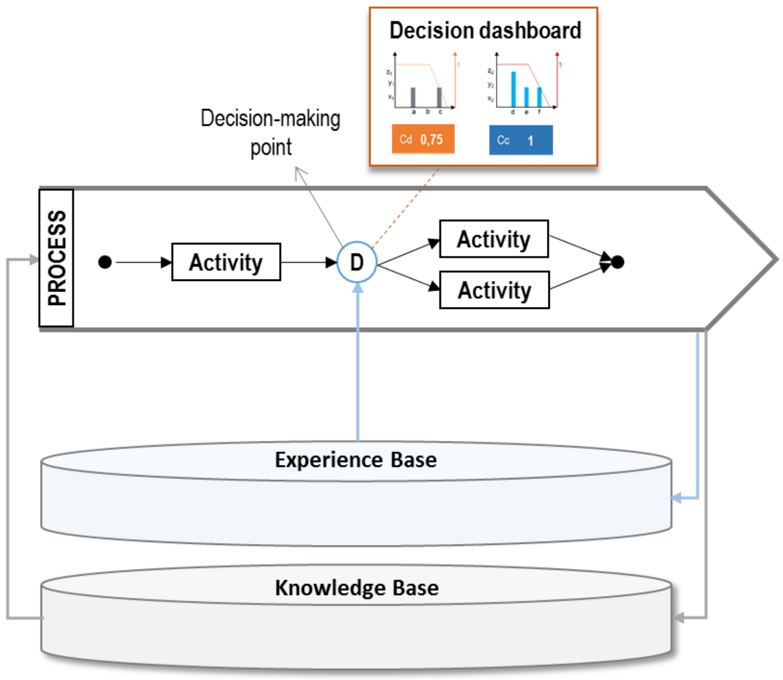


Figure 16: Agile process illustration

Decision-making points are also key elements of an agile process in our model. A decision allows comparing a set of feasible options and to choose the most convenient one to solve a problem (Alvarado et al., 2005). Moreover (Kast, 1993) claims that a decision problem contains four elements: the set of possible choices, the set of uncontrolled elements (which represents uncertainty), the set of possible consequences and the relation between decision, uncontrolled elements and consequences. According to (Malakooti, 2012), decision making can be defined through the following steps: 1) Problem formulation, 2) Problem solving technique, 3) Multi-criteria decision making (selection of the best option) and 4) Decision realization (action to be performed).

In the proposed agile process, two kinds of decision-making points are used: “nominal” and “based on event” decision-making points. During the execution of an agile process, every **nominal decision-making point** determines a gate to decide what to do next in the process. When a decision is reached, decision makers take into consideration and evaluate available information regarding the process, summarized in a decision dashboard, in order to define which option of the process will be executed next. Decision makers can, for instance, choose among performing an activity that leads to the alternative 1, or another activity that leads to the alternative 2.

In order to help decision making, at each decision-making point, a compilation of information from previous experiences, represented in a *decision dashboard,* is proposed to users. Dashboards along with key performance indicators provide quick and precise information (Peral et al., 2017), which is the objective of an efficient help to decision making.

On the other hand, **decision-making points based on event** are unplanned. When an unexpected event occurs, a decision-making point based on event is created in order to decide what to do next. The decision based on event can lead to modify the structure of the process to overcome the unexpected situation. If the structure of the process is modified, it will be stored in the Experience Base as an "event-scenario" and will receive special treatment for its future reuse (see section 4.2).

Nominal decision-making points are defined in a first version of the process (V_0) which is created before the process execution by using existing processes in the knowledge base. The mechanism which enables to create versions of the process, including the first version (V_0), is described in the next section.

3.3 Agile process versioning system

The agile process model is composed of incremental versions. At every decision-making point, a new version of the process (V_{n+1}) is created. The first version of the process model (V_0) includes activities and nominal decision-making points. It is built from existing processes stored in the Knowledge Base (see section 3.5). In the Knowledge Base, a set of options to perform a given type of process (e.g. "Problem-solving" process) is available. This set of options is the baseline for the first version of the process. The process manager (see section 3.4) is supported by previous process information compiled in a decision dashboard in order to decide if something needs to be added or modified to the proposed first version V_0.

A new version of the process is created any time there is a change from the previous version. Three typical situations during the execution of the process lead to the creation of a new version:

1. At each decision-making point, an option is chosen among the set of proposed options in order to continue the process. Then, the process is changed due to this choice and a new version is created.
2. The process can be modified when activities or decision-making points are added or modified in order to better respond to changes or problems. Then, a new version is created.
3. When an unexpected event occurs, if activities and/or decision-making points are added, a new version of the process is also created.

The versioning of the agile process goes from V_0 to V_n. V_n is the last version of the process (i.e. referred to the last decision-making point or to the last modification included in the process). Every new version includes a notation mentioning the result of decisions. This notation builds a trace of all the decisions made during the process along with the rationale that led to that decision. This versioning system facilitates the formalization of experiences, their capitalization and their future reuse.

The versioning system of an agile process has been presented. In the next section, in order to define and execute an agile process the necessary roles are described.

3.4 Roles within the agile process

Taking into consideration agile practices (mostly from agile software development methods) referring to team empowerment, three roles are proposed in this approach in order to manage and execute an agile process. However, the definition of the roles does not intend to restrict the application of the model. Thus, several roles may be assumed by the same person if necessary (e.g. in a small company). The three roles defined for the agile process are presented in Table 3.

Process manager	S/he is in charge of the characterization and of the definition of the first version of the process. S/he can also manage the execution of the process. The process manager has sufficient knowledge regarding the process itself. For instance, for a problem-solving process the process manager could be the quality/continuous improvement manager.
User	Any person that will use the process during its execution. It is considered that all users have a minimal level of knowledge concerning the process. It is important to notice that in the case where a regular user has sufficient understanding about the process itself, s/he can take the process manager role.
Knowledge expert	S/he defines process models that are added into the knowledge base either from scratch, or from a prior performed process. S/he is in charge of validating knowledge (e.g. knowledge contained within an experience).

Table 3: The roles within the agile process model

It is important to specify that "decision makers", a term used along this work, is the group of people involved in the decision-making activity. Then, all of the roles could be part of the group depending on their participation during the process. For instance, if the process manager is in charge of the whole process, s/he is the sole decision maker. On the other hand, if a group of users are given with all the responsibilities of the process, they are the decision makers.

The roles of an agile process have been presented in this section. In order to ensure knowledge and experiences capitalization and their future reuse, dedicated mechanisms are introduced in the next section.

3.5 Knowledge and Experience-Driven Process

In the proposed approach, an agile process is based on knowledge and experiences capitalization and reuse principles. In order to introduce the dedicated capitalization and reuse mechanisms, the necessary elements are presented in this section.

The aim of this section is to define the basis related to knowledge and experiences in order to define the lifecycle of the agile process in chapter 4. The knowledge and experience bases are described in sections 3.5.1 and 3.5.2. Experiences tagging system, which enables a fast retrieval mechanism, is presented in section 3.5.3. The process indicators used within dashboards are defined in section 3.5.4.

3.5.1 The Knowledge Base

The Knowledge Base (KB) is a structured collection of pieces of knowledge. In this study, every piece of knowledge corresponds to a specific standard process. Then, the KB is a repository composed of processes. All processes contained in the knowledge base are classified, according to their nature, in different kinds of Types of Processes (TP). The process classification and the definition of the types of processes may change from a company to another one. In the “Cross industry process classification framework”(“Process Classification Framework | APQC,” 2015) thirteen categories of high-level processes are defined including several processes corresponding to each category. A selection of a sample of ten processes of the “Develop and Manage Business Capabilities” category has been done in this study in order to define a baseline of types of processes to be contained in the Knowledge Base. The selected processes, illustrated in Figure 17, are: Improve processes, Manage projects, Establish quality requirements, Evaluate performance to requirements, Manage non-conformance, Implement and maintain the enterprise quality management system, Assess knowledge management capabilities, Create and manage organisational performance strategy, Evaluate process performance, and Train and educate functional employees. These high-level types of processes can be used to structure the knowledge base, even if other types can be defined. For instance, in this work, the problem-solving type of process is added (TP: Problem Solving).

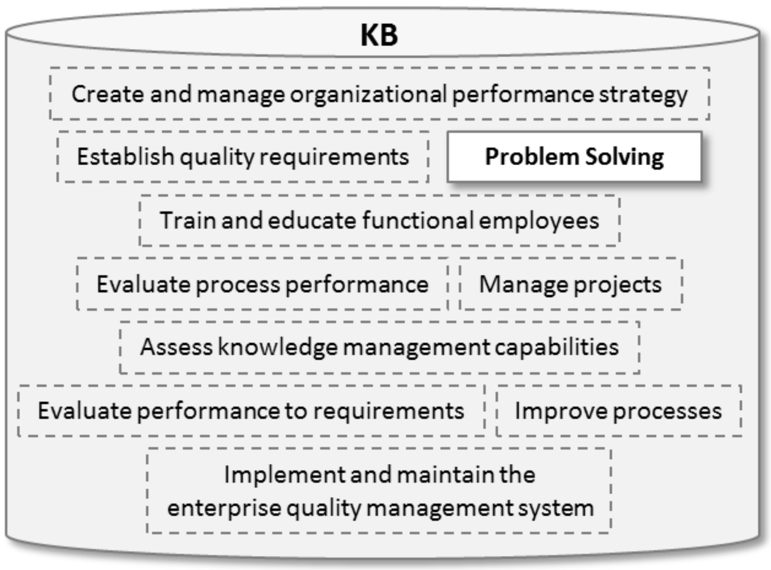

KB
Create and manage organizational performance strategy
Establish quality requirements
Problem Solving
Train and educate functional employees
Evaluate process performance
Manage projects
Assess knowledge management capabilities
Evaluate performance to requirements
Improve processes
Implement and maintain the enterprise quality management system

Figure 17: Example of types of processes (TP) composing the KB, inspired from ("Process Classification Framework | APQC," 2015)

Every type of process is composed of one or more scenarios. Each scenario corresponds to a different manner to perform a process, referring to changes in its structure. It means that every structural change on a process creates a new scenario which has to be stored in the knowledge base. Structural changes considered in this study are:

- Modification of the sequence of activities,
- Modification of the description of the activity,
- Addition of one or more activities.

For instance, consider the type of process "Problem Solving", and the 8D process within problem solving (the 8D process corresponds to a sub-category of the type of process Problem Solving). The 8D process can be represented by the eight standard sequential activities. In that case, there is only one possible scenario. However, it can be decided to define other alternative where two activities can be performed in parallel. Therefore, a second scenario is possible. Figure 18 represents the 8D process model with the two possible scenarios (1) and (2). Activities are represented with rectangles and denoted with A plus a number.

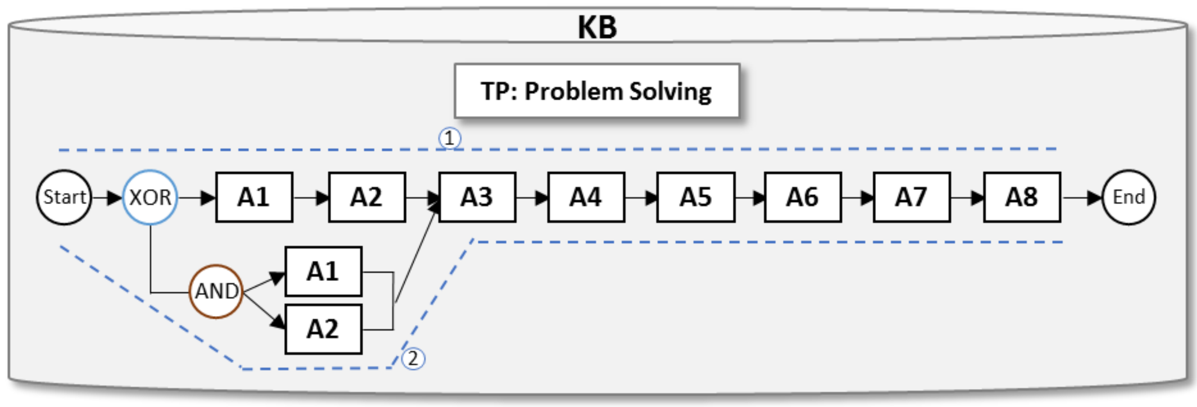


Figure 18: Illustration of two process scenarios for a same type of process

During process execution, a choice between scenarios 1 and 2 needs to be done, before performing activity A1. The choices between different scenarios stored in a knowledge base constitute the first pillar of an agile process.

Consequently, the knowledge base is composed of different types of processes. As illustrated in Figure 19, each type of process is represented by an oriented and acyclic graph. The nodes of the graph are XOR, AND or scenario activities. The edges represent the precedence constraints between the nodes. Such edges can be identified with an ID, which is used in order to distinguish which action was taken after a decision (see section 4.1.4). *Scenario* activities allow describing a process and are stored into the Knowledge base. The XOR nodes allow the representation of possible options and then, to define scenarios. The AND nodes permit to represent the parts of the process which can be performed in parallel. Therefore, the graph enables the representation of all scenarios corresponding to a type of process in a simplified manner, mainly since XOR nodes allow grouping generic activities that are common to two or more scenarios. Nodes have a specific attribute called “list of events” that is used to treat unexpected events during the process execution (see section 4.2). The UML diagram of Figure 19 represents the components of the graph for a type of process.

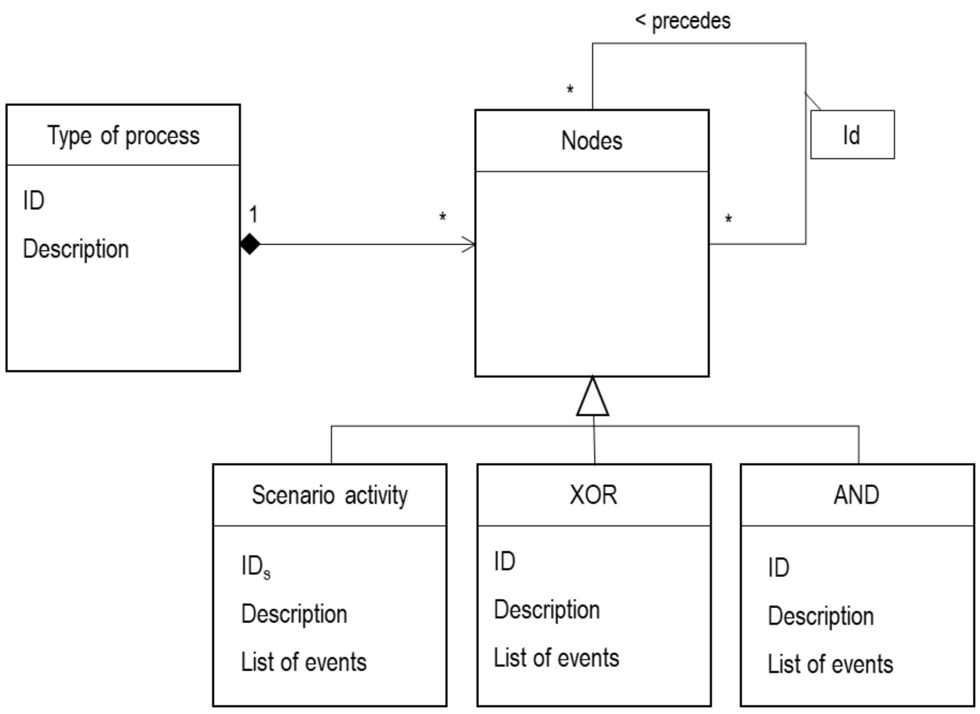


Figure 19: Type of process UML representation

The storage of processes into the knowledge base can be achieved in two different manners. First, at the creation of the knowledge base (i.e. when the KB is empty) it may be considered that knowledge can be defined from international recognized standards (e.g. such as EN 9136 (IAQG (International Aerospace Quality Group), 2016) for 8D/9S in the Aerospatiale domain), company rules or best practices. In this case, the knowledge expert validates the knowledge and adds the new process into the KB, into its corresponding type of process. Otherwise, knowledge can be created from experiences. After the execution of the process, if its structure does not exist in the knowledge base (i.e. there is no scenario that matches completely the structure of the new process), it is the responsibility of the knowledge expert to validate (or not) the new process as knowledge and to store it on the knowledge base.

3.5.2 The Experience Base

From the knowledge base, for a given type of process, the scenarios which have been performed/executed are called *experiences* and are stored into the Experience Base (EB). For this purpose, in the EB, a scenario gathers a set of experiences that present the same structure and it corresponds to the scenario that exists in the KB. Experiences correspond to instances of the execution of scenarios. This means that every time that a process scenario is performed, a new experience linked to the scenario is added to the experience base. For the scenario (1) illustrated in Figure 18, three experiences are represented (Figure 20).

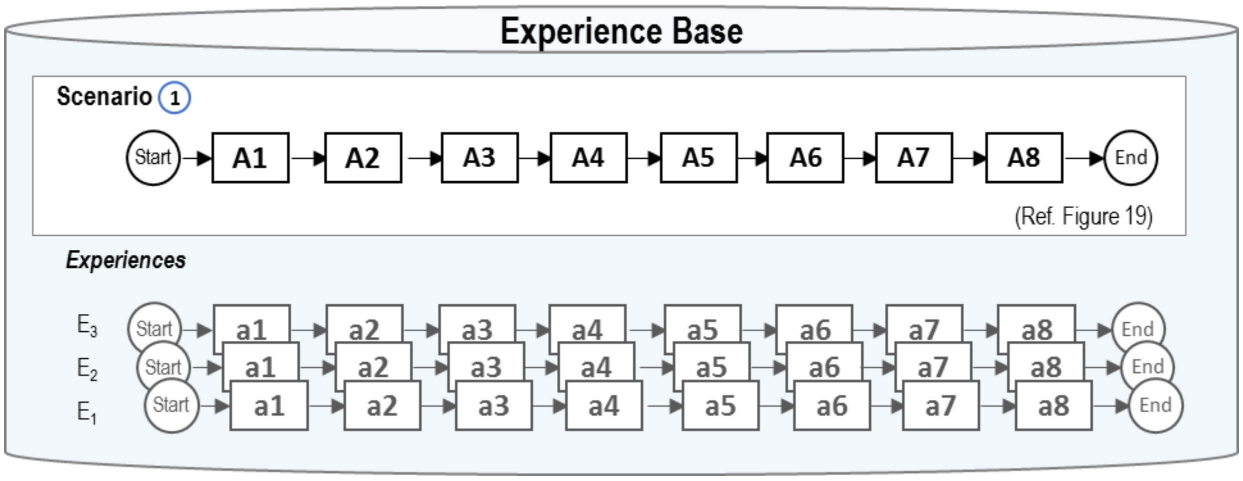


Figure 20: Illustration of three experiences for a given scenario

Activities are the building blocks of every process and, consequently, of each experience. In order to allow decision support through decision dashboards, activities are given with indicators (cost and delay) during the execution of the process. *Generic* activities are activities included in the KB representation, considered as "general" since they are not given with a set of proper indicators. An experience activity is an instance of a scenario activity of the type of process in the knowledge base, stored in the Experience base. Also, experiences are given with global cost and delay indicators. In Figure 21, type of process, scenario and experience are illustrated with an UML representation including the relations between these entities. Types of processes are associated to scenarios that are themselves associated to experiences. Experiences have tags as attributes and are composed of activities that are characterized by individual cost and delay values. Tags are described in section 3.5.3 and cost and delay indicators in section 3.5.4.

Therefore, each experience is modelled by an oriented and acyclic graph which contains only activity nodes and precedence constraints (the edges). Two fictive activities are used: "start" and "end". Their duration and cost are equal to zero.

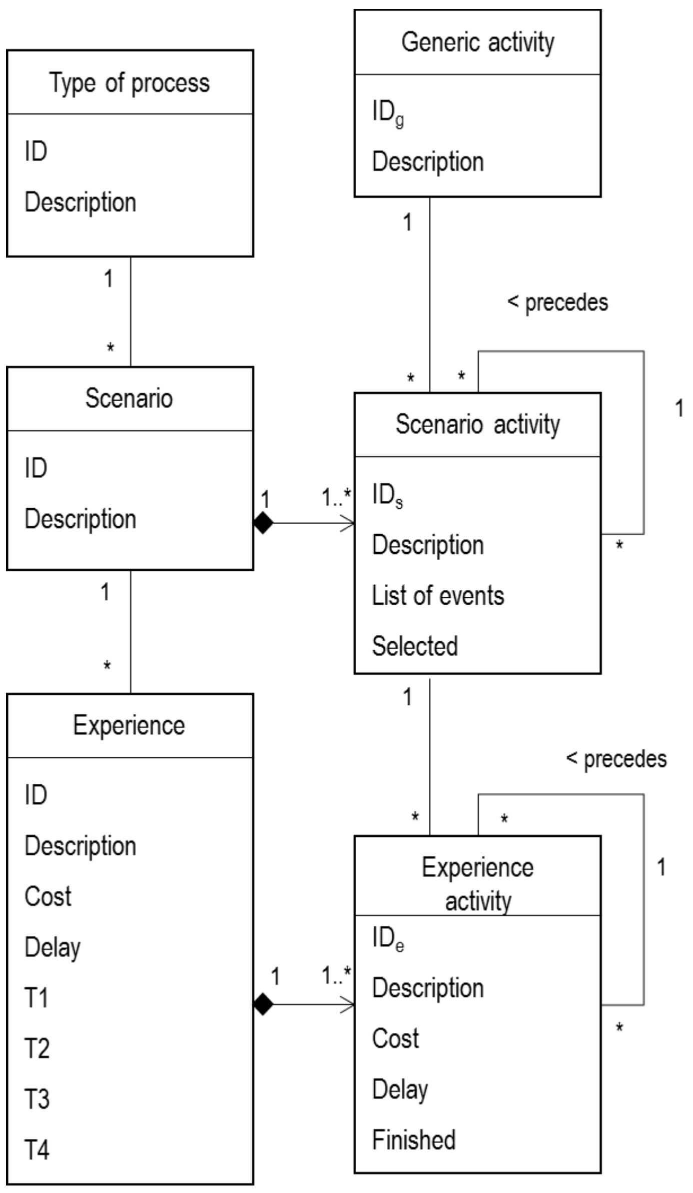


Figure 21: UML representation of type of process, scenario, experience and activity

3.5.3 Tagging system

In order to index an experience, a set of tags is defined and linked to the experience. Each set of tags synthesizes the most important characteristics of the experience. The set of tags is identified during the definition of the process (see section 4.1.1) and is employed in order to find similar past experiences (see section 4.1.2).

In our approach, tags are considered as concepts and they are gathered within taxonomies and stored into the knowledge base. Taxonomies are considered in this work for being a simplified ontology (McGuinness, 2002). An ontology is "a formal, explicit specification of a shared conceptualization" (Gruber, 1993). A taxonomy is a hierarchical structure described through relations between concepts included in the hierarchy (Van Rees, 2003). Taxonomies create a coherent representation of concepts through their structuration into a tree according to their similarity (Jabrouni, 2012). In our work, taxonomies are defined for each type of tag in order to characterize the process and to enable a research into the experience base. To retrieve similar past experiences semantic similarities are used (see section 4.1.2).

For the purpose of this research, considering that a focus on problem-solving processes is done, the four tags described in Table 4 are defined for each experience. However, our approach is generic and other kind of tags can be used.

Process Model (T1)	It refers to the standard or procedure that is used to define the process. For instance, for problem-solving processes, process models could be 8D, 9S, PDCA, DMAIC methodologies.
Enterprise department (T2)	It refers to the sector of the enterprise where the process belongs to (e.g. Manufacturing, finance, human resources, marketing, etc.).
Type of product (T3)	It refers to the type of product concerned by the problem. Let us consider a bike factory, some examples of products could be the frame, saddle, front, wheels, or pedals.
Type of Problem (T4)	It characterizes the problem that is being solved. For instance, painting, dimensional, material, or assembling problem.

Table 4: Agile problem-solving process tags

Then, in a scenario S_i, for an experience E_{ij}, the set of tags is: $ToE_{ij}= <T1_{ij}, T2_{ij}, T3_{ij}, T4_{ij}>$.

An example of simplified taxonomies that will be used further in this paper is shown in Figure 22. The example is based on a bicycle factory. The taxonomy represented in Figure 22a corresponds to the process models regarding problem-solving processes such as 8D, PDCA and DMAIC. In Figure 22b enterprise departments' tags are represented. In Figure 22c, the tags corresponding to the type of product concerned by the problem are represented. Four tags corresponding to types of problems are shown in Figure 22d.

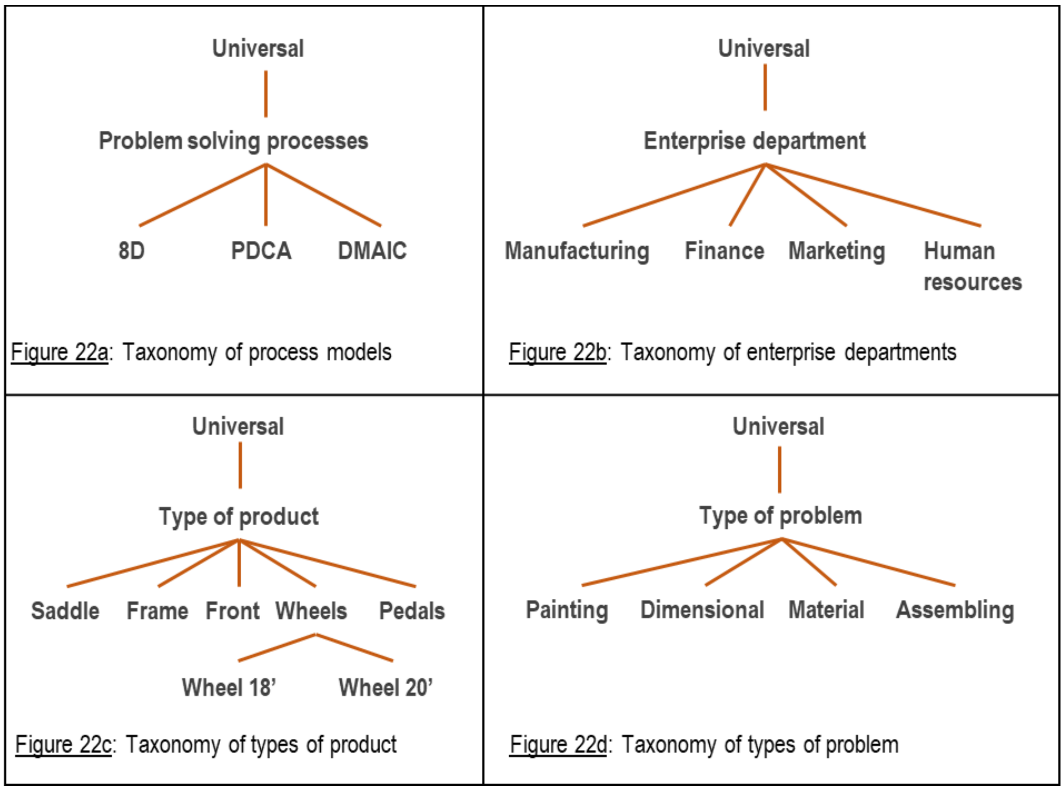


Figure 22: Example of simplified taxonomies

3.5.4 Process indicators

In this study, indicators are computed for all the activities within a process. When the process is finished, the experience is stored in the experience base along with its indicators. When defining a new process, the available experiences indicators are used to help decision making using dashboards.

During the process execution, at the end of each activity, its cost and delay values are computed, then, global indicators can be computed for the experience. Moreover, global indicators for scenarios are computed in order to compare different scenarios during decision making (see section 4.1.4). It means that when a new process is being defined, the set of indicators from previous experiences, grouped into scenarios and stored in the experience base, can help to decide which scenario to choose.

For the purpose of this study, the indicators represented in the dashboard are the probability distributions of cost and delay values for all possible scenarios for a given process type. Each scenario corresponds to one or more experiences, and each experience is given with a cost and delay value. Then, the probability distributions of all experiences values of cost and delay for a given scenario are represented on the decision dashboard. Moreover, a set of requirements in terms of cost and delay is defined according to the objectives that the process should satisfy (see chapter 4 for more details). The global compatibilities with regards to those requirements are also represented in the dashboard. During the execution of the process, the information presented on the decision dashboard is updated according to the indicators regarding already performed and remaining activities. The computing of the decision dashboard is detailed in section 4.1.4.

3.6 Illustration of an agile process

In order to illustrate the concepts, an agile process is illustrated in this section through an example. In Figure 23, the knowledge base containing an agile process is presented. For the type of process "problem solving", all available scenarios are retrieved in the knowledge base. Then, for each scenario, one or more associated experiences may exist in the experience base.

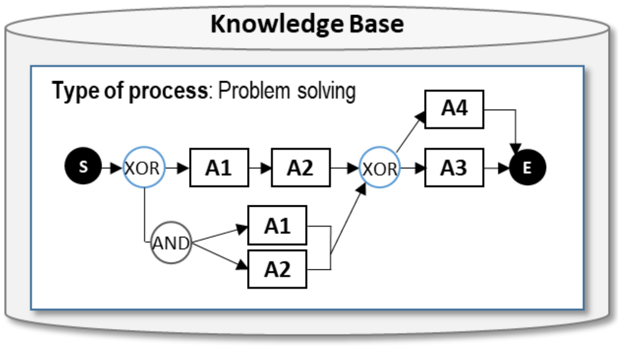


Figure 23: Illustration – Knowledge base for the type of process: problem solving

For the chosen type of process in the knowledge base, the first version of the process is built. XOR nodes from the knowledge base are replaced by decision-making points in the process first version. The first version of the process, illustrated in Figure 24, shows the available options to perform the process. For instance, in the first decision point (D1) it can be chosen to perform the activities A1 and A2 in parallel or sequentially, and in the second decision point (D2), it can be chosen to perform either A3 or A4. The four available scenarios are illustrated in the lower part of Figure 24.

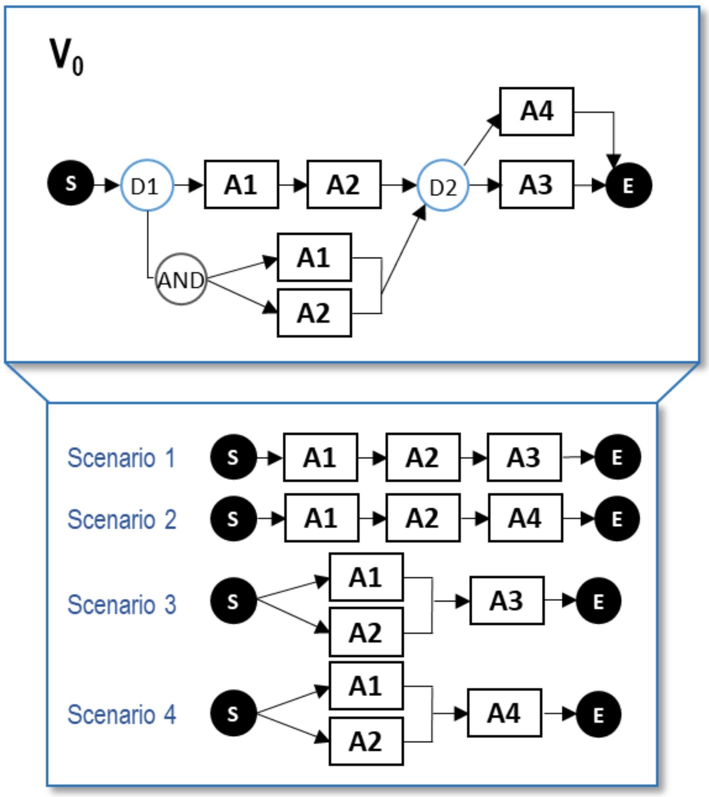


Figure 24: Agile process first version

During process execution, users consider the process first version as a baseline in order to carry out the process. At each decision-making point, a dashboard is proposed in order to help decision making. Such a dashboard includes cost and delay probability distributions and compatibility in order to show how compatible such values are compared to the target ones. The dashboard construction is detailed in section 4.1.4.

Let us consider that decision makers take into consideration the decision dashboard in order to execute the process. The evolution of the process, explained through its versions, is represented in Figure 25. For instance, in D1 the decision was to perform the activity A1 and then A2. This first decision is visible in the second version of the process (V1), as illustrated in Figure 25b. The same reasoning is applied to create the third version of the process (V2), illustrated in Figure 25c. Here, the decision (D2) was to perform A3 instead of A4. Once the process was completed, the new experience needs to be capitalized. Considering V2, the last version of the process, the structure of the process (A1-A2-A3) matches an existing scenario (Scenario 2 illustrated in Figure 24). The experience is then capitalized into the existing scenario.

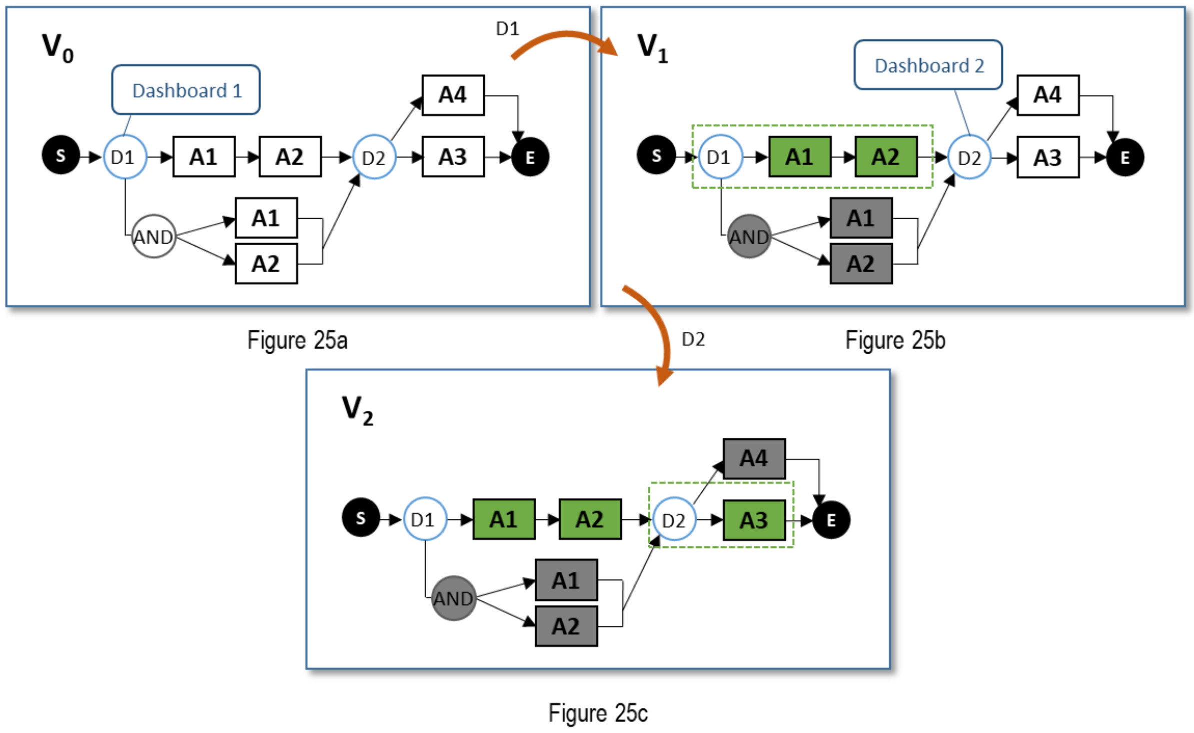


Figure 25: V_0, V_1 and V_3 of the agile process

3.7 Conclusion

The information model for agility was presented in this chapter. The agile process proposed in this work was introduced and its main key elements were outlined. The system that allows recording of the changes that occur during the execution of an agile process (versioning system) was introduced. The roles that are needed to define and execute such a process were described.

The knowledge and experience bases were presented along with the elements that are stored. The notions of types of processes, of scenario and experience were explained. Finally, the tagging system and the indicators defined in order to characterize an agile process were introduced.

The proposed illustration enabled understanding of the principles behind an agile process.

This chapter permitted the unification and understanding of the concept "agile process". However, some questions arise: How is this process defined? How is it executed? And, how is it retrieved from the experience base? Next chapter allows answering of those questions through the definition of the agile process lifecycle.

4. AGILE LIFECYCLE MODEL

The constituent elements of an agile knowledge and experience based process are defined in chapter 3, setting the information model of this work. The model proposed in this research, considering the definition of an agile problem-solving process model, is completed with a dynamic cycle that allows to create and to execute such a process. In order to guide the definition and the execution of an agile problem-solving process a specific *lifecycle* is proposed. The lifecycle introduced in this chapter allows the incorporation of knowledge and experience capitalization and reuse principles, from the definition of the agile process, its execution, until its storage in the experience and knowledge bases once the process is over.

Furthermore, the treatment of unexpected events is introduced at the end of this chapter. Our agile process permits to define decision-making points and to react to disturbances.

The agile lifecycle is described in the first part of this chapter (section 4.1). The treatment of unexpected events is detailed in section 4.2.

4.1 The agile process lifecycle

The proposed agile lifecycle is inspired by Case-based Reasoning (CBR) principles (Aamodt and Plaza, 1994), previously detailed in chapter 2. It is composed of five steps deployed around the knowledge and experience bases, as illustrated in Figure 26. The first step "Process scope definition" consists in defining and characterizing the current process. The "Experience filtering" step aims to apply filters to select the most similar past experiences according to the current process information. This step can be compared to the "retrieve" phase of CBR. "Adaptation of the first version of the process" considers the first version proposed from the knowledge base and, if necessary, modified by the process manager. It can be compared to the "reuse" phase of CBR. "Process execution/continuous adaptation" refers to the execution of the agile process. Finally, "Storage in EB/KB" is performed when the agile process is over, in order to collect all instances of the process in the Experience Base and, if necessary, in the Knowledge Base. This step can be compared to the "retain" phase of CBR.

Each one of the agile lifecycle phases along with an illustrative example is presented in the following sections.

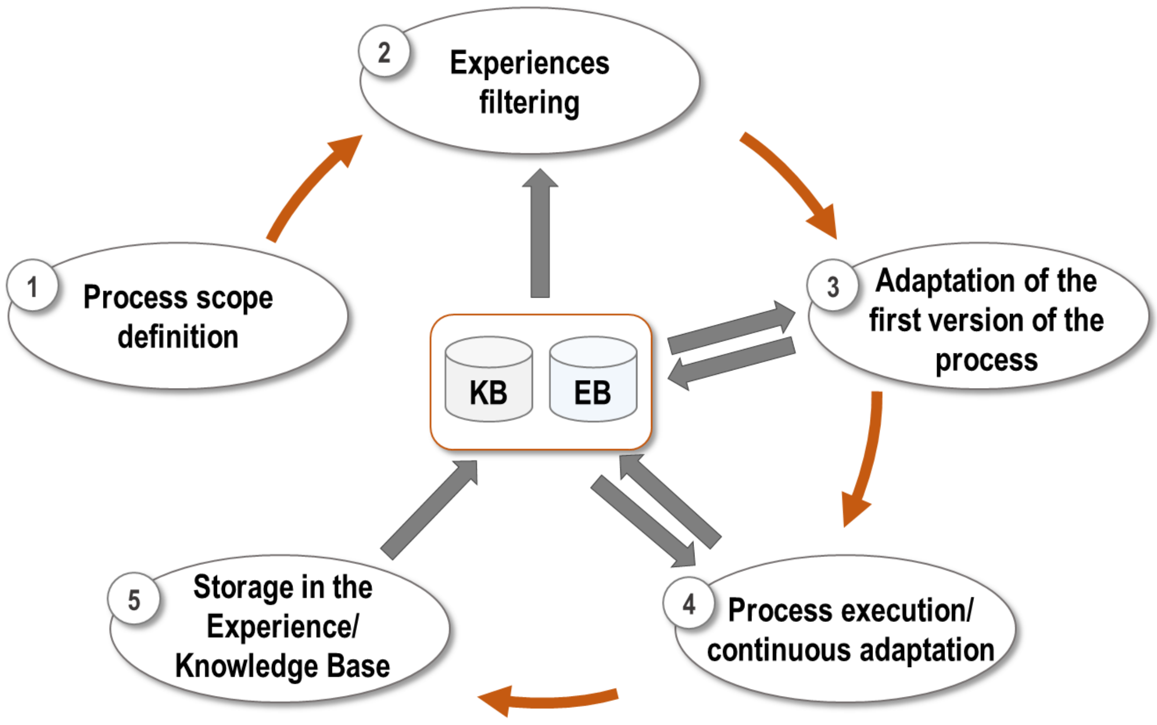

2
Experiences filtering
1
Process scope definition
KB
EB
3
Adaptation of the first version of the process
5
Storage in the Experience/ Knowledge Base
4
Process execution/ continuous adaptation

Figure 26: Agile lifecycle

4.1.1 Step 1. Definition of the process scope

The objective of the first step is to describe and to characterize the current process which has to be carried out. This characterization of the process aims to define its characteristics and constraints, useful data that will be used to perform the experience filtering phase (section 4.1.2). This step is performed by the process manager who is in charge of the entire process lifecycle. It means that the process manager describes the process and gathers all necessary information from stakeholders. According to the Cambridge dictionary a stakeholder is "a person such as an employee, customer, or citizen who is involved with an organisation, society, etc. and therefore has responsibilities towards it and an interest in its success" (Cambridge University Press, 2017). We refer to stakeholders as all actors directly involved and/or impacted by the results of the process. It includes the external and internal customers (quality manager, accounts, etc.), project manager, etc.

For the purpose of this study, two stages are proposed in order to define the process scope. The first step strives for characterizing the process. The second one aims at gathering all the process constraints from stakeholders. Both steps are detailed next, followed by the illustrative example.

4.1.1.1 Definition of the process context

This stage aims at establishing all the characteristics that define the agile process. Two major activities are performed during the definition of the process context. In a first time, from the list of types of processes capitalized in the knowledge base (see section 3.5.1), the type of the current process is selected by the process manager. The type of process list can be modified for a specific company, the objective being that the list includes all the company's types of processes which may be defined and executed through the agile lifecycle.

The second activity to be performed is the definition of a set of tags which allows formalizing the complete description of the process following the requirements. As it has been detailed in section 3.5.2, a set of tags is defined for each process in order to synthesize its most important characteristics. Four mandatory tags need to be defined for an agile problem-solving process: Process Model (T1), Enterprise department (T2), Type of product (T3), and Type of Problem (T4). The set of tags identified for the current process is denominated "Target tags" (denoted by Tt = {Tt_1, Tt_2, Tt_3, Tt_4}). It is important to notice that more tags could be added in order to improve precision in the experience filtering phase (see section 4.1.2), such as: Expert involved in the process (T5), the required competencies of the expert (T6), etc. However, this study is only focused on the four previously mentioned tags.

The target tags are used as an input of step 2/ "Experience filtering", to search and filter similar past experiences in the experience base (see section 4.1.2). This means that the set of target tags is compared to the set of tags linked to each experience stored in the experience base, searching semantic similarity. For this purpose, the expected level of similarity between the target tags and the experience tags (in the experience base) needs to be defined. Two cases are proposed:

a) It can be decided that an experience tag must be equal to its corresponding target tag (i.e. both tags are equal). Then, the similarity between the target and the experience tags must be equal to one. This case is used when the process manager decides that s/he wants to retrieve only past experiences that respect the equality of certain tags. For instance, if the process manager wants to retrieve from the experience base only 8D processes, the similarity must be equal to one between the target tag Tt_1=8D and the experience tag T_1=8D. In this case, the attribute of the tag indicates that it is a non-flexible tag. Then, the nature of the tag T_i is *non-flexible* and is denoted as *T_i.nature= Non-Flexible*.
b) If there is no need to retrieve the same tag value, the similarity can be computed and characterized by a numerical value. For instance, the process manager wants to perform an 8D process but s/he wants to see all problem solving experiences that are available in the experience base. Then, the target tag Tt_1 is equal to "8D" and it is compared to the experience tag T_1 with a computed similarity for each experience. In this case, the attribute of the tag T_i indicates that it is a flexible tag. Then the nature of the tag is *flexible* and is denoted as *T_i.nature= Flexible*.

At this stage, a threshold Gt has to be defined. It is used during the computation of the global similarity between Tt and a set of tags linked to an experience. If this global similarity measure is inferior to the threshold, the experience is not used (section 4.1.2).

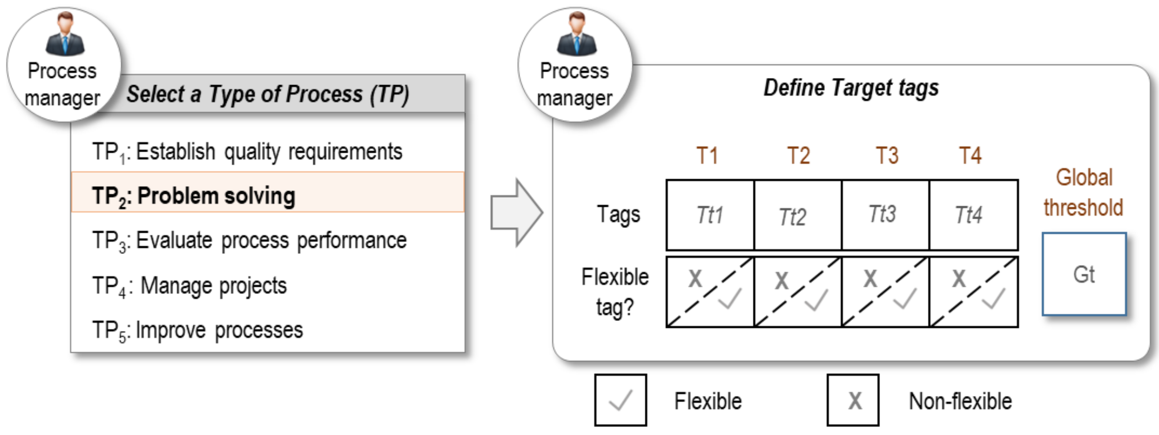


Figure 27: Definition of the process context

Then, when the process manager selects the type of process as illustrated in the left part of Figure 27 (the "Problem solving" type of process is selected), s/he identifies the set of tags and defines whether each tag is flexible or not, as illustrated in the right part of Figure 27. Also, the global threshold for the flexible tags needs to be defined at this stage. It is important to notice that, if during experience filtering step, no experience match the required similarity level, the process manager can decide to modify the threshold in order to include more experiences and evaluate possible deviations during decision making.

Once the process is completely characterized through its set of tags, the stakeholders need to define the constraints to be respected during the execution of the process.

4.1.1.2 Definition of process constraints

All actors interested and/or involved in the process have different constraints regarding cost, delay, involved resources or others. The purpose of this step is to collect process constraints from stakeholders.

Two constraints are considered in this study, the process cost and the process delay. First, the Cost (C) of the process refers to the maximum cost allowed to perform the process. Second, the Delay (D) regards the maximum duration allowed for the execution of the process. The values of C and D set by stakeholders allow to constraint the process. This means that, when retrieving similar past experiences, constraints are considered in order to compare them with prior experiences. This is possible since cost and delay indicators are measured and recorded for each activity within a process, as well as the global process cost and delay indicators (see section 3.5.4).

Then, during this step, the process manager collects all cost and delay constraints along with their owner (i.e. the stakeholder that set the constraint) and defines the global values of C and D.

Furthermore, the process manager considers that cost and delay constraints are flexible. All constraints are considered as fuzzy constraints (Dubois et al., 1996). This allows more flexibility when past experiences indicators do not satisfy current process constraints but their values are close to the target ones. Then, constraints release may be negotiated with the corresponding stakeholder (see section 4.1.4).

Based on the fuzzy model of a soft constraint proposed by (Dubois et al., 1996), the soft constraints C (cost) and D (delay) are described through the following equations. Let TC be the Target Cost and TD be the Target Delay. The parameters α and β allow the definition of flexibilities within the soft constraints C and D.

The soft constraints are defined by means of their respective compatibility functions (equations 1 and 2). These functions allow to define how a value *c* (respectively *d*) of cost (respectively of delay) is compatible with the soft constraints of cost C (respectively the soft constraints of delay D).

$$Compat_{\alpha}(c,TC) = \begin{cases} 1 & if \;\; c \leq TC \\ \frac{(1+\alpha)*\mathrm{TC}-\mathrm{c}}{\alpha*\mathrm{TC}} & if \; TC < c \leq (1+\alpha)*\mathrm{TC} \\ 0 & otherwise \end{cases} \quad (1)$$

$$Compat_{\beta}(d,TD) = \begin{cases} 1 & if \;\; d \leq TD \\ \frac{(1+\beta)*\mathrm{TD}-\mathrm{d}}{\beta*\mathrm{TD}} & if \;\; TD < d \leq (1+\beta)*\mathrm{TD} \\ 0 & otherwise \end{cases} \quad (2)$$

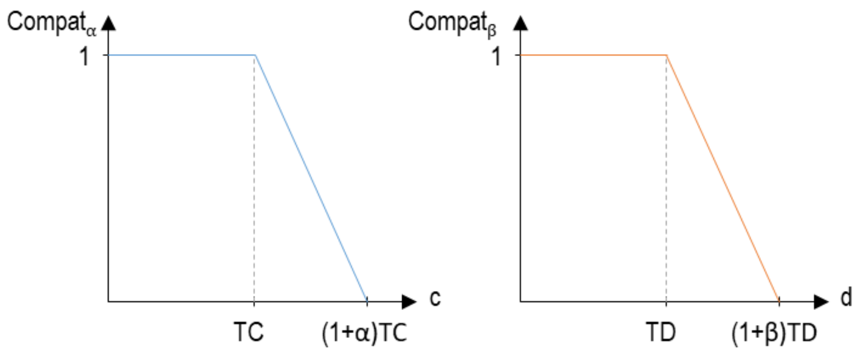


Figure 28: Cost and delay compatibility functions

4.1.1.3 Illustrative example

In order to illustrate the definition of the process scope step, an illustrative example is provided. Let us imagine that a bicycle manufacturing company deploys a simple process to treat their simple problems. For the purpose of this example, let us consider a problem detected in the assembly line, regarding non-conformance painted pedals. The quality department needs to solve this problem and to avoid its recurrence. The process manager designated for this problem-solving process is the quality manager.

During the first step, the process manager characterizes the process scope. First, the set of target tags are defined. For this purpose, the chart illustrated in Figure 29 is completed.

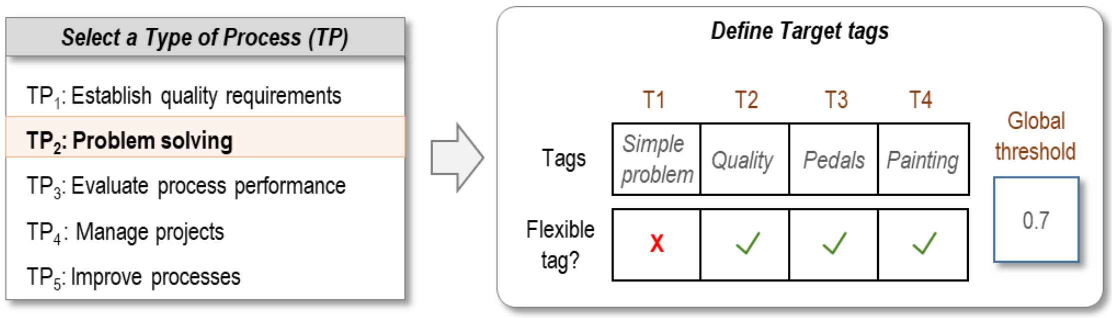


Figure 29: Definition of the process context - Illustration

The targeted type of process is “Problem solving”, the required process model (represented by the tag T1) is “simple problem” (the method used within the company), all problems treated by the quality department (tag T2), referring to the pedals (tag T3), and concerning the painting (tag T4). The tag T1 is defined as non-flexible, it means that only past processes performed through the “simple problem” method should be retrieved. The other tags are flexible. Then, a global threshold of 0.7 is defined.

Also, constraints need to be defined during this step. For this purpose, the process manager collects information from stakeholders. The first stakeholder (a client affected by this problem) demands that the problem is solved within 3 days. The second stakeholder, the manufacturing manager demands that the process cost is less than 1000€. The process manager sets tolerance levels in order to set the constraints as flexible. As illustrated in Figure 30, defined values are for cost, α=0.1 ((1 + 0.1) x 1000 = 100) and for delay β = 0.33 ((1 + 0.33) x 3 = 4).

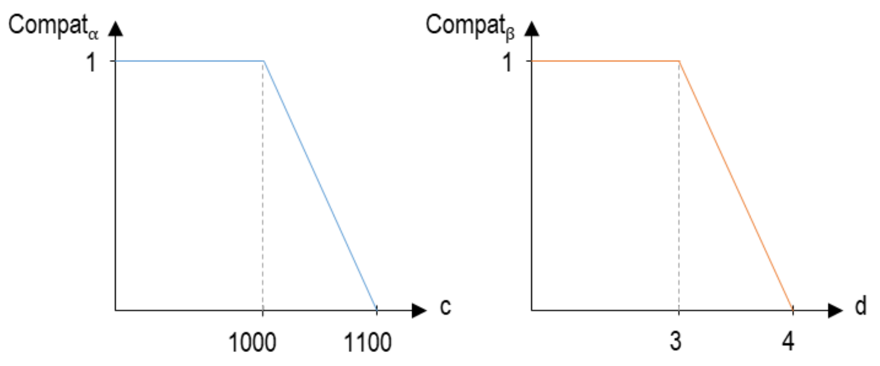


Figure 30: Cost and delay compatibility functions - Illustration

At the end of this step, the process is characterized through the definition of its set of target tags and its cost and delay constraints. Then, considering the collected information, the experience filtering step can be performed. Experience filtering is detailed in the next section.

4.1.2 Step 2. Experience filtering

Using as inputs the set of target tags defined during the process context definition step (section 4.1.1.1), the experience and knowledge bases are consulted in order to get similar past experiences and knowledge. When experiences tags available in the experience base match the target tags, such experiences are taken into consideration to define the first version of the process (V_0).

The similarity between the set of target and experience tags needs to be measured. This is possible through measuring semantic similarity between tags that belong to the same taxonomy (Jabrouni et al., 2011). It means that, for a given experience each tag is compared to its corresponding target tag, and their similarity is computed (i.e. T1 is compared to Tt_1, T2 to Tt_2, etc.). The measure of similarity of Wu and Palmer, (Wu and Palmer, 1994) is chosen for this study because of its simplicity and efficiency (3):

$$Sim\ (t_1, t_2) = \frac{2 * \mathrm{depth}(t_{com})}{\mathrm{depth}(t_1) + \mathrm{depth}(t_2)} \qquad (3)$$

Such that:

- $\mathrm{depth}(t_i)$ is the number of nodes on the path from the root concept to t_i,

- t_{com} is the first common ancestor of t_1 and t_2 in the taxonomy.

Using this formula, and depending on the nature of target tags (flexible or non-flexible), experiences are filtered. The process defined to perform the filtering phase is illustrated in Figure 31 and detailed below. Furthermore, the experience filtering algorithm is described in Algorithm 1.

2.1) The first step represented in Figure 31, consists in using a simple query in order to find, from the selected type of process available in the knowledge base, all its associated scenarios in the experience base. For each scenario, a set of experiences exists in the experience base (see Figure 21 in section 3.5.2).

2.2) The second step allows the computation of the global similarity of each selected experience corresponding to the scenarios associated to the required type of process.

2.3) In the third step, the similarity value of non-flexible tags is considered. The objective of defining a tag as non-flexible is to retrieve only the same value of the tag, this means that similarity between both tags must be equal to one. Then, results different to one should be excluded from the selection since they are not interesting. It means that, when similarity between a target non-flexible tag and an experience tag is different to one, the experience is discarded and not furthered considered during the current agile process lifecycle. If, on the other hand, the value of similarity is equal to one for each non-flexible tag, the experience is saved and the remaining tags are analysed to verify that the global similarity is larger than the global threshold.

Once experiences have been filtered according to their non-flexible tags, the set of flexible tags is considered for a second filtering (or if all tags are flexible a first filtering is performed at this stage) represented in step 2.4 of Figure 31. For a given experience, the individual values of similarity between its flexible tags and the target tags computed in step 2.2, are considered. Then, the global similarity (named Sim_g) of the experience is computed using a GOWA operator (Yager, 2004) described in the equation 4. The parameter β permits to tune the operator from minimum to maximum function. If $\beta \rightarrow +\infty$, the GOWA operator functions as a Maximum. If $\beta \rightarrow -\infty$, it functions as Minimum. The value β=2 corresponds to a quadratic mean, etc.

$$Sim_g = (\sum_{k=1}^{4} \frac{1}{4} \times Sim(Tt_k, t_k)^{\beta})^{1/\beta} \qquad (4)$$

The global similarity value Sim_g should, at least, be equal to the defined global threshold Gt. It means that, if the global similarity value is inferior to the threshold, the experience is discarded. On the other hand, if it is equal or superior, it is selected. Then, the same process is repeated for all experiences of all scenarios. Algorithm 1 allows performing these four steps.

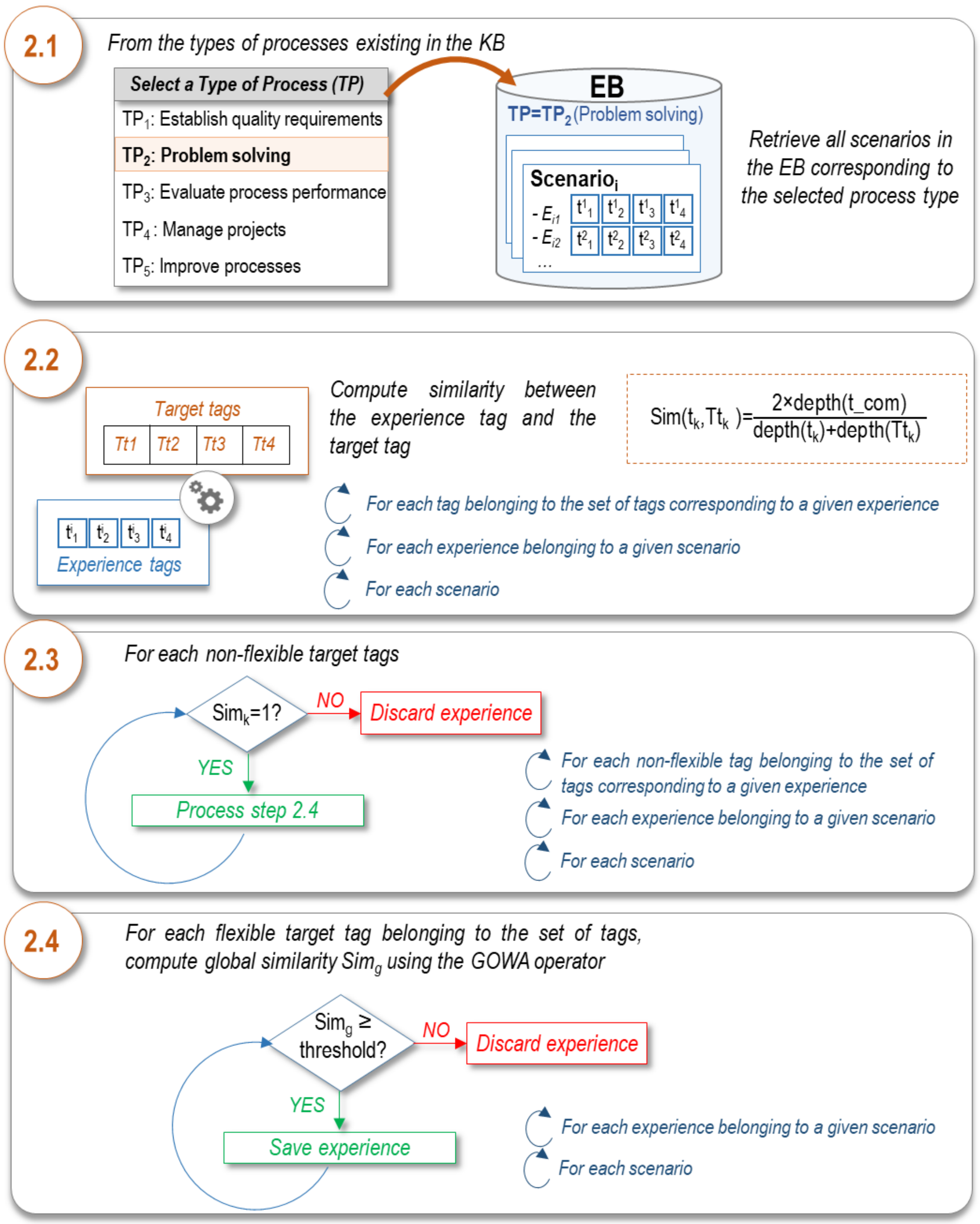

2.1
From the types of processes existing in the KB
Select a Type of Process (TP)
TP1: Establish quality requirements
TP2: Problem solving
TP3: Evaluate process performance
TP4: Manage projects
TP5: Improve processes
EB
TP=TP2 (Problem solving)
Scenario_i
- E_i1
- E_i2
Retrieve all scenarios in the EB corresponding to the selected process type
2.2
Target tags
Tt1 Tt2 Tt3 Tt4
Experience tags
Compute similarity between the experience tag and the target tag
Sim(t_k,Tt_k) = 2×depth(t_com) / (depth(t_k)+depth(Tt_k))
For each tag belonging to the set of tags corresponding to a given experience
For each experience belonging to a given scenario
For each scenario
2.3
For each non-flexible target tags
Sim_k=1?
NO
Discard experience
YES
Process step 2.4
For each non-flexible tag belonging to the set of tags corresponding to a given experience
For each experience belonging to a given scenario
For each scenario
2.4
For each flexible target tag belonging to the set of tags, compute global similarity Sim_g using the GOWA operator
Sim_g ≥ threshold?
NO
Discard experience
YES
Save experience
For each experience belonging to a given scenario
For each scenario

Figure 31: Experience filtering process

```
Start
Declare variables $S_i$, $S_{TP}$, $E_{ij}$, $SE_i$, $t_k$, $ToE_{ij}$, $Tt_k$, Tt, ExpKO, $SE'_i$, Sum, Nb
Initialize variables ExpKO = FALSE, $SE'_i$ = Ø, Sum = 0, Nb = 0
For each $S_i \in S_{TP}$
    For each $E_{ij} \in SE_i$
      For each $t_k \in ToE_{ij}$
        Calculate $Sim_k = Sim(Tt_k, t_k) = \frac{2 * \text{depth}(t_{com})}{\text{depth}(Tt_k) + \text{depth}(t_k)}$
        If ($Tt_k$.nature = Non Flexible and $Sim_k \neq 1$)
           ExpKO = TRUE
           Break
        End If
      End For
      If ExpKO = FALSE then
         For each $Tt_k \in Tt$
          If ($Tt_k$.nature = Flexible) then
               $Sum = Sum + (Sim_k)^{\beta}$
               Nb = Nb+1
          End If
         End For
         $Sum = \frac{Sum}{Nb}$
         $Sim_g = \sqrt[\beta]{Sum}$
         If ($Sim_g \geq$ threshold or Nb = 0) then
               $SE'_i = SE'_i \cup E_{ij}$
         End If
      End if
    End For
End For
End
```

S_i:	Scenario i
S_{TP}:	Set of scenarios corresponding to the selected type of process
E_{ij}:	Experience j of the scenario i
SE_i:	Set of experiences corresponding to S_i
t_k:	Tag k corresponding to a set of tags ToE_{ij}
ToE_{ij}:	Set of tags of the experience E_{ij}
Tt_k:	Target tag k
Tt:	Set of target tags
Tt_k.nature:	Nature of the Tt_k
Sim_k:	Similarity between t_k and Tt_k
ExpKO:	Experience discarded (yes/no)
Sim_g:	Global similarity of flexible tags
SE'_i:	Filtered set of experiences

Algorithm 1: Algorithm to filter experiences from the set of scenarios

4.1.2.1 Illustrative example

In this section, the example introduced in section 4.1.1.3 is further developed

Let us consider that the scenarios illustrated in the upper part of Figure 32 are those available in the knowledge base for the type of process "Problem solving". In order to perform step 2.1, these available scenarios are retrieved in the experience base, as represented in the lower part of Figure 32. Let us consider that five scenarios are retrieved. The activities represented in the problem-solving processes of Figure 32 are: A0: Define the problem; A1: Apply immediate containment actions; A2: Perform root cause analysis; A3: Define and implement corrective actions; A4: Define preventive actions; A5: Standardize and transfer the knowledge.

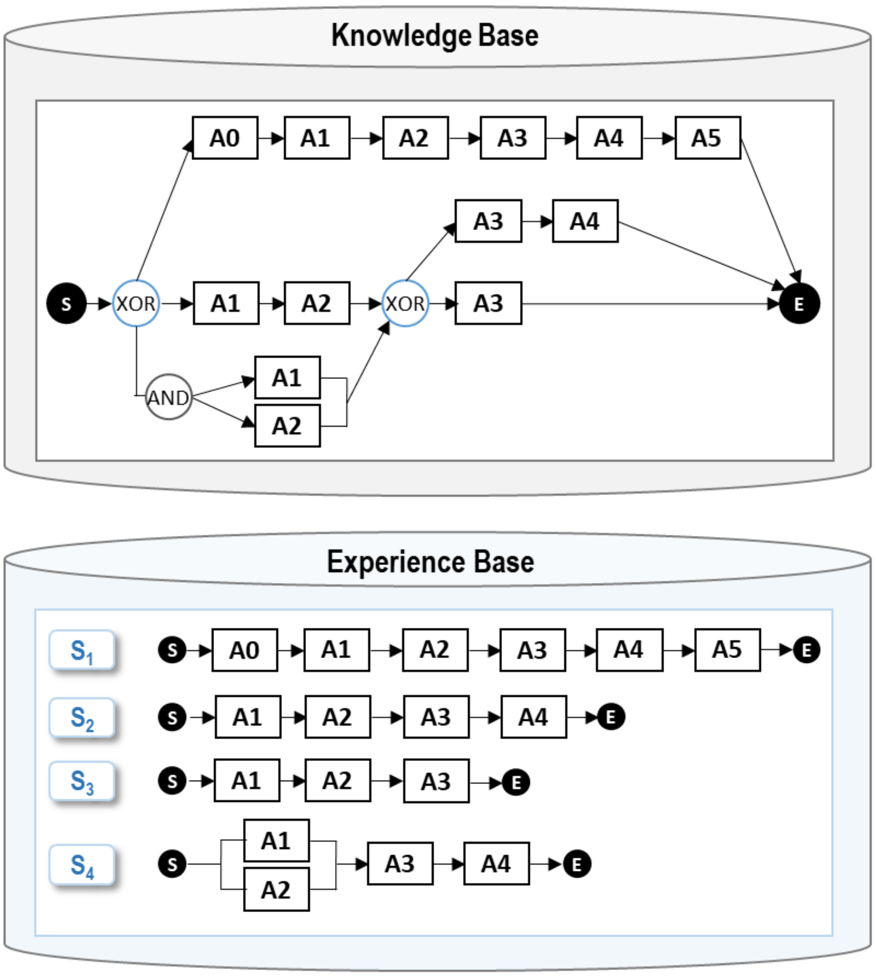


Figure 32: Available scenarios for TP: Problem solving

The experience filtering for each scenario is performed next. For this purpose, the sets of tags corresponding to each experience are presented in Table 5. For the purpose of this study, one experience is linked to scenario S_1, three to S_2, two to S_3, and four experiences to S_4. The first number on the experience's notation refers to the scenario and the second to the experience. It is important to notice that the scenario S_5 that exists in the knowledge base (A1 and A2 in parallel, and A3) is not considered in the experience base since it has never been performed before, therefore, there are no experiences associated to the scenario.

Moreover, Figure 33 illustrates the taxonomies of tags. The taxonomy for process model tags (used for tag T1) is illustrated in Figure 33a, two types of problem-solving processes are available: Simple and complex problems. In Figure 33b, the taxonomy for enterprise departments (used for T2) is illustrated. In Figure 33c, the taxonomy for products (used for T3) is presented. The problems taxonomy (used for T4) is illustrated in Figure 33d.

Scenario	Experience	T1	T2	T3	T4
S_1	E_{11}	Complex problem	Quality	Saddle	Painting
S_2	E_{21}	Simple problem	Quality	Frame	Dimensional
	E_{22}	Simple problem	Manufacturing	Pedals	Painting
	E_{23}	Simple problem	Finance	Saddle	Material
S_3	E_{31}	Simple problem	Quality	Front	Painting
	E_{32}	Simple problem	Manufacturing	Saddle	Assembling
S_4	E_{41}	Simple problem	Quality	Frame	Painting
	E_{42}	Simple problem	Quality	Wheels	Painting
	E_{43}	Simple problem	Manufacturing	Pedals	Painting
	E_{44}	Simple problem	Manufacturing	Pedals	Material

Table 5: Experiences tags – Illustration

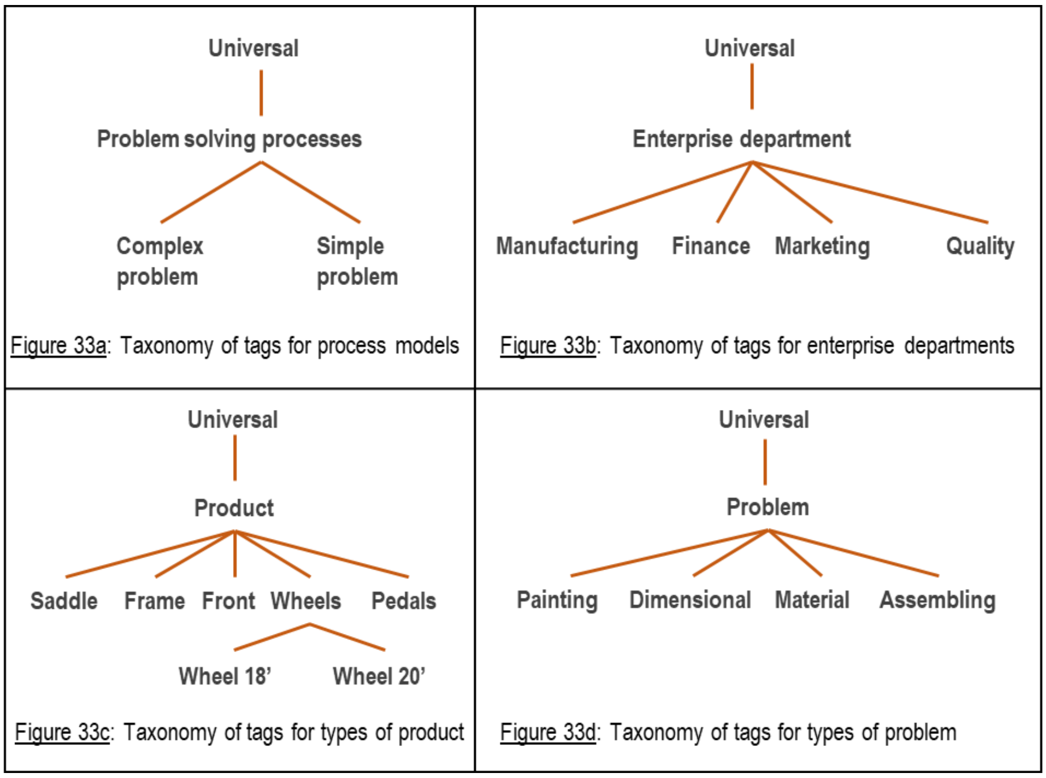


Figure 33a: Taxonomy of tags for process models

Figure 33b: Taxonomy of tags for enterprise departments

Figure 33c: Taxonomy of tags for types of product

Figure 33d: Taxonomy of tags for types of problem

Figure 33: Taxonomies of tags - Illustration

Following step 2.2 of Figure 31, the similarity between each tag of the experiences and target tags is computed with formula (3) and the taxonomies of Figure 33. All similarity values are presented in Table 6.

		T1	T2	T3	T4	Global similarity (Sim_g)
	TT	SIMPLE	QUALITY	PEDALS	PAINTING	
	Flexible/Non Flexible	Non Flexible	Flexible	Flexible	Flexible	
S_1	E_{11}	Complex	Quality	Saddle	Painting	*N/A*
		Sim=0.67	*Sim=1*	*Sim=0.67*	*Sim=1*	
S_2	E_{21}	Simple	Quality	Frame	Dimensional	0.80
		Sim=1	*Sim=1*	*Sim=0.67*	*Sim=0.67*	
	E_{22}	Simple	Manufacturing	Pedals	Painting	0.90
		Sim=1	*Sim=0.67*	*Sim=1*	*Sim=1*	
	E_{23}	Simple	Finance	Saddle	Material	0.67
		Sim=1	*Sim=0.67*	*Sim=0.67*	*Sim=0.67*	
S_3	E_{31}	Simple	Quality	Front	Painting	0.90
		Sim=1	*Sim=1*	*Sim=0.67*	*Sim=1*	
	E_{32}	Simple	Manufacturing	Saddle	Assembling	0.67
		Sim=1	*Sim=0.67*	*Sim=0.67*	*Sim=0.67*	
S_4	E_{41}	Simple	Quality	Frame	Painting	0.90
		Sim=1	*Sim=1*	*Sim=0.67*	*Sim=1*	
	E_{42}	Simple	Quality	Wheels	Painting	0.90
		Sim=1	*Sim=1*	*Sim=0.67*	*Sim=1*	
	E_{43}	Simple	Manufacturing	Pedals	Painting	0.90
		Sim=1	*Sim=0.67*	*Sim=1*	*Sim=1*	
	E_{44}	Simple	Manufacturing	Pedals	Material	0.80
		Sim=1	*Sim=0.67*	*Sim=1*	*Sim=0.67*	

Table 6: Similarity values for all tags - Illustration

Step 2.3 (Figure 31) considers non-flexible tags. In this example, T1 is a non-flexible tag. Considering similarities of Table 6, the only experience with similarity different to one for T1, is E_{11}. This experience is discarded.

During step 2.4 (Figure 31), the global similarity Sim_g is computed for each experience using the GOWA operator. This global similarity is compared to the threshold in order to filter the experiences. The global similarity values are presented in the last column of Table 6. In this example, the similarities of experiences E_{23} and E_{32} do not satisfy the threshold of 0.7 therefore, they are discarded.

Then, after discarding experiences E_{23} and E_{32}, the filtered scenarios corresponding to this agile process are obtained and are represented in Figure 34 (i.e. scenarios S_2, S_3, S_4).

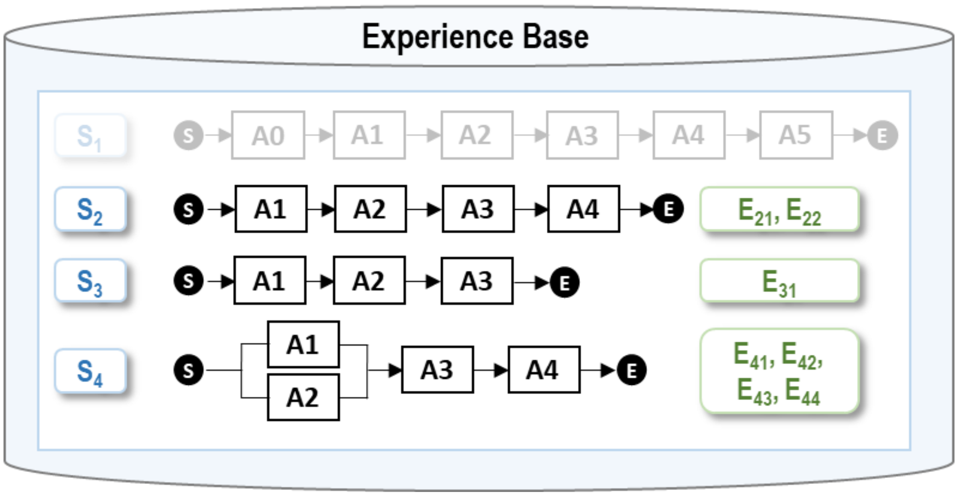


Figure 34: Filtered scenarios for the current lifecycle

At the end of this step, only experiences with tags sufficiently similar to the current process are kept. In the next step, these filtered experiences are used to help the process manager to adapt the first version of the process to the current process needs. They constitute valuable information useful to help decision makers to make choices and to be agile.

4.1.3 Step 3. Adaptation of the first version of the process

During this step, the first version (V_0) of the process (see section 3.3) extracted from the knowledge base is adapted by the process manager (see Figure 35) in order to allow agility during the process execution. This first version of the process is an instance of the corresponding process model found in the knowledge base. Every XOR node is transformed into a decision-making point that allows deciding which path to perform during the execution of the process (section 4.1.4).

The complete process model is considered with the purpose of ensuring agility during process execution, since all possible paths should be visible for decision makers.

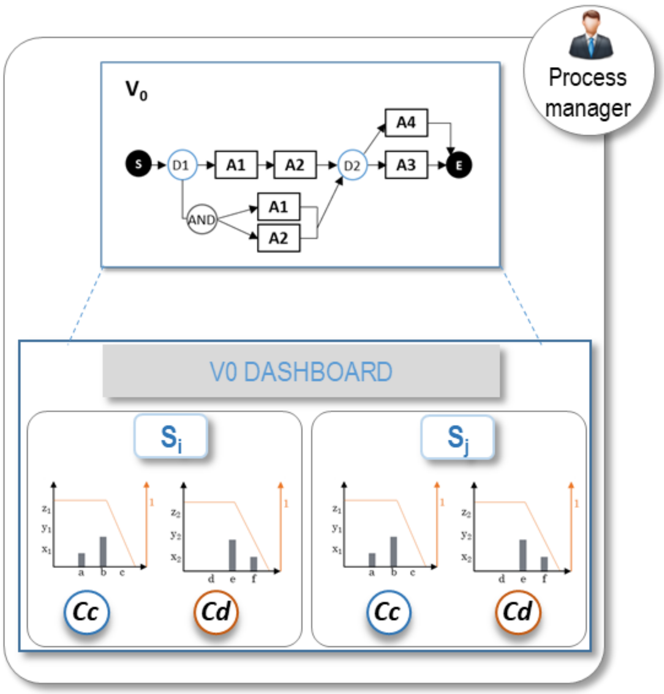


Figure 35: Adaptation of the first version of the process

Then, filtered scenarios and experiences (i.e. output of the experience filtering step), relevant information and indicators are compiled into a decision dashboard denominated *V_0 dashboard,* as illustrated in the lower part of Figure 35. V_0 dashboard is proposed to the process manager in order to help decision making regarding the adaptation of the first version of the process. During this step, the process manager can choose to modify the proposed V_0 according to the V_0 dashboard and/or personal criteria (i.e. s/he can decide to create a new scenario). Moreover, the process manager can choose a “preferred scenario” in V_0 that will be proposed to users in order to carry out the process. The preferred scenario can be defined from indicators of V_0 dashboard or from the process manager personal criteria.

The combination of the first version of the process and of the dashboard allows the process manager to understand experiences performed in the past and how they match the current process. Only scenarios that contain one or more experiences are visible in V_0 dashboard since its purpose is to provide information about past experiences. It means that, in the cases that a scenario was discarded during experience filtering step (i.e. all of its experiences were discarded), or if the scenario is the result of a combination of some parts of previously performed scenarios but it has never been carried out, in both cases the scenario is not represented in the dashboard.

The adaptation of the process first version is illustrated in Figure 36. The steps referred to the elaboration of the V_0 dashboard indicators are detailed next.

V_0 dashboard

The V_0 dashboard is constructed from *filtered* scenarios and experiences (section 4.1.2). After the experience filtering step, each scenario contains a filtered set of experiences. Then, for each filtered scenario, a compatibility distribution of the cost and delay values corresponding to all filtered experiences is computed. For this purpose, first, the probability distributions of cost and delay values in the filtered scenario are computed. Afterwards, the compatibility of each value with regards to the soft constraints C and D is calculated. The objective is to provide global compatibility values of cost and delay for a given scenario, and to graphically illustrate them. The set of values that cost and delay indicators can take when performing a given scenario and how compatible those values are compared to the constraints are represented. This information is presented into the V_0 dashboard in order to help the process manager to adapt V_0. Two aggregated indicators (one for cost and one for delay) are then computed and shown in the dashboard.

Indicators presented in V_0 dashboard are: cost and delay probability distributions, global cost and delay compatibility values (Cc and Cd), number of experiences of the scenario before filtering (nexp0), number of experiences of the scenario after filtering (nexp1).

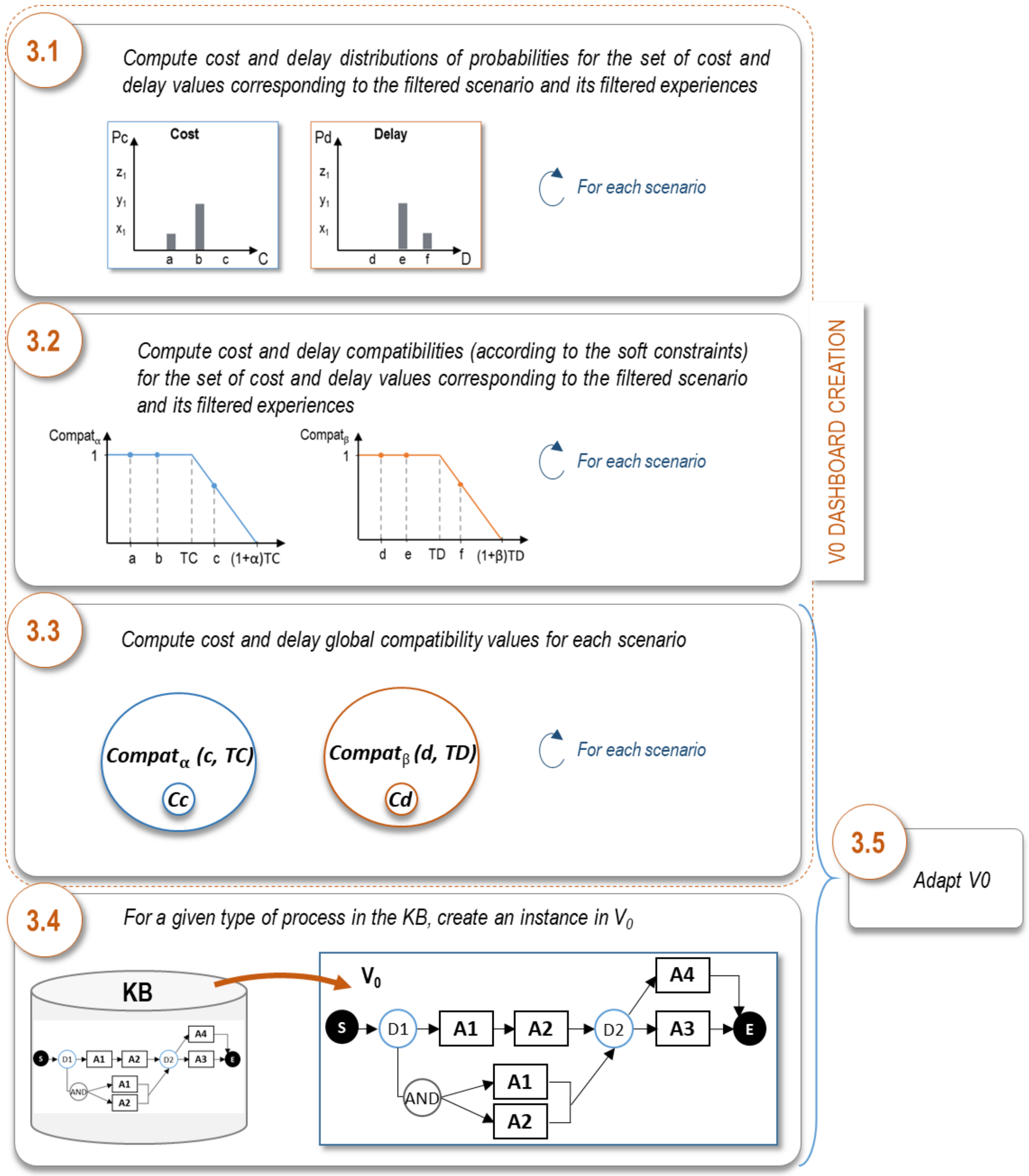

3.1
Compute cost and delay distributions of probabilities for the set of cost and delay values corresponding to the filtered scenario and its filtered experiences
Pc
Cost
z1
y1
x1
a
b
c
C
Pd
Delay
d
e
f
D
For each scenario
3.2
Compute cost and delay compatibilities (according to the soft constraints) for the set of cost and delay values corresponding to the filtered scenario and its filtered experiences
Compatα
1
a b TC c (1+α)TC
Compatβ
d e TD f (1+β)TD
For each scenario
V0 DASHBOARD CREATION
3.3
Compute cost and delay global compatibility values for each scenario
Compatα (c, TC)
Cc
Compatβ (d, TD)
Cd
For each scenario
3.5
Adapt V0
3.4
For a given type of process in the KB, create an instance in V0
KB
V0
S
D1
A1
A2
D2
A4
A3
E
AND
A1
A2

Figure 36: Adaptation of V_0

Step 3.1: Algorithm 2 allows the computing of the cost and delay distributions of probabilities for each scenario i.

```
Start
Declare variables $S_i$, $S_{TP}$, $E_{ij}$, $SE'_i$, $D_i$, $C_i$, $d_{ij}$, $c_{ij}$, $c_k$, $d_k$, $Nbck$, $Pc_{ki}$,
                  $Nbd_k$, $Pd_{ki}$
Initialize variables $D_i = 0$, $C_i = 0$, $Nbc_k = 0$, $Pc_{ki} = 0$, $Nbd_k = 0$, $Pd_{ki} = 0$
For $S_i \in S_{TP}$
    For $E_{ij} \in SE'_i$
      $D_i = D_i \cup d_{ij}$
      $C_i = C_i \cup c_{ij}$
    End For
    For $c_k \in C_i$
          For $E_{ij} \in SE'_i$
            If ($c_k = c_{ij}$) then
              $Nbc_k = Nbc_k + 1$
            End If
          End For
        $Pc_{ki} = \frac{Nbc_k}{|SE'i|}$
    End For
    For $d_k \in D_i$
          For $E_{ij} \in SE'_i$
            If ($d_k = d_{ij}$) then
            $Nbd_k = Nbd_k + 1$
            End If
          End For
        $Pd_{ki} = \frac{Nbd_k}{|SE'i|}$
    End For
  $Nbc_k = 0$
  $Nbd_k = 0$
 End For
End
```

S_i: *Scenario i*
S_{TP}: *Set of scenarios corresponding to the selected type of process*
E_{ij}: *Experience j of the scenario i*
SE'_i: *Filtered set of experiences for scenario i*
D_i: *Set of different delays of the experiences of SE'_i*
C_i: *Set of different costs of the experiences of SE'_i*
d_{ij}: *Delay of the experience j of the scenario i*
c_{ij}: *Cost of the experience j of the scenario i*
Nbc_k: *Number of occurrences of c_k*
Nbd_k: *Number of occurrences of d_k*
Pc_{ki}: *Probability that c_k exists within SE'_i*
Pd_{ki}: *Probability that d_k exists within SE'_i*

Algorithm 2: Compute of cost and delay distribution of probabilities for V_0 dashboard

Step 3.2: The compatibility between each cost and delay value with respect to constraints is computed. For this purpose, functions defined in section 4.1.1.2 for fuzzy constraints are considered (equations (1) and (2) are recalled below). Then, compatibility is computed with equations (1) and (2) for the set of cost and delay values described in the previous step. If a value (cost or delay) is inferior or equal to the target (TC or TD), then this value is fully compatible (compatibility=1). When the cost or delay value is between the target and the defined limit value, the compatibility is given by the function described in equation (1) and (2) (between 0 and 1). If cost or delay values are superior to the limit value, the compatibility is zero.

$$Compat_{\alpha}(c, TC) = \begin{cases} 1 & if\ \ c \leq TC \\ \frac{(1+\alpha)*\mathrm{TC}-\mathrm{c}}{\alpha*\mathrm{TC}} & if\ TC < c \leq (1+\alpha)*\mathrm{TC} \\ 0 & otherwise \end{cases} \tag{1}$$

$$Compat_{\beta}(d,TD) = \begin{cases} 1 & if \;\; d \leq TD \\ \frac{(1+\beta)*TD-d}{\beta*TD} & if \;\; TD < d \leq (1+\beta)*\mathrm{TD} \\ 0 & otherwise \end{cases} \qquad (2)$$

Step 3.3: The global compatibility of cost and delay is computed for each scenario. In this study, the global compatibility is computed as the sum, for each value, of the product of the probability of the value in the filtered scenario and the compatibility of the value with regards to the soft constraint (equations (1) and (2)). Then, considering the outputs of Algorithm 2 and of equations (1) and (2), the two values of cost and delay compatibility are computed for each scenario. The V_0 dashboard is constructed considering the filtered scenarios, the probability distributions, the compatibilities, and the global compatibility values, for cost and delay.

The global compatibility of the values of costs for scenario i with regards to the target cost TC is given by:

$$Comp(C_i,TC) = \sum_{c_k \in C_i} Pc_{ki} * Compat_{\alpha}(c_k,TC) \qquad (5)$$

The global compatibility of the values of delays for scenario i with regards to the target delay TD is given by:

$$Comp(D_i,TD) = \sum_{d_k \in D_i} Pd_{ki} * Compat_{\beta}(d_k,TD) \qquad (6)$$

In the dashboard, the global compatibility value for cost is named *Cc*, and for delay *Cd*.

Furthermore, probability distributions and compatibilities are graphically represented in the dashboard, as illustrated in Figure 37. The vertical bars indicate the probability of each value, for instance in Figure 37, the probability for *a* is 0.25; 0.5 for *b* and 0.25 for *e*. The function (blue line) represents the compatibility of each value with regards to the soft constraint. For instance, in Figure 37, the function takes the value one for *a* and *b*, and 0.5 for *e* (it can be computed through equation (1)).

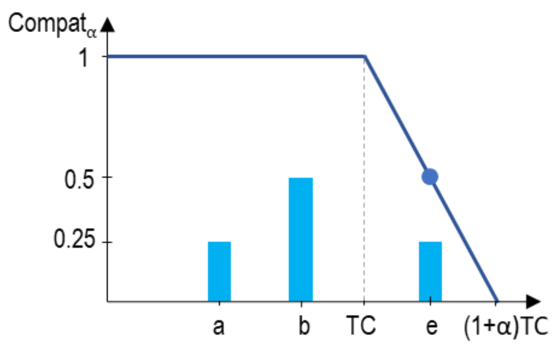


Figure 37: Example of probability distribution and compatibility function for cost

Step 3.4 of Figure 36 illustrates the extraction of all scenarios corresponding to a type of process from the knowledge base. XOR nodes in the knowledge base are replaced by decision-making points in V_0. For a given process type, all of its scenarios are used to construct V_0. The process manager, after consulting V_0 dashboard, may decide to modify these scenarios. For the purpose of this study, only the addition of new scenarios is allowed. Scenarios cannot be deleted since it would decrease flexibility during the execution of the process. Even if the scenario does not match cost and/or delay target values, it should be available for consultation during process execution.

Then, considering outputs of step 3.3 and 3.4, the process manager can adapt the process first version (**step 3.5**). V_0 dashboard is constructed for the example detailed in the next section.

4.1.3.1 Illustrative example

The first version of the process is adapted in this step. For this purpose, first, the probability of the cost and delay values in the filtered scenarios is computed. The cost and delay values for the filtered experiences are presented in Table 7.

Scenario	Experience	c (€)	d (days)
S_2	E_{21}	900	3.5
	E_{22}	1000	2
S_3	E_{31}	1000	3
S_4	E_{41}	1100	3
	E_{42}	1000	5
	E_{43}	900	4
	E_{44}	1000	3.5

Table 7: Cost and delay values corresponding to filtered experiences

In the second column of Table 8, the probabilities of cost and delay values within each scenario are presented. In the third column, the value of compatibilities with regards to the target values is presented. Finally, in the fourth column, the global compatibilities cost and delay values for each scenario are presented.

Scenario	Probability	Compatibility	Global compatibility
S_2	Pc(900)=0.5	C_α(900;1000)=1	Comp(C_{s2};1000)=1
	Pc(1000)=0.5	C_α(1000;1000)=1	
	Pd(3.5)=0.5	C_β(3.5;3)=0.5	Comp(D_{s2};3)=0.75
	Pd(2)=0.5	C_β(2;3)=1	
S_3	Pc(1000)=1	C_α(1000;1000)=1	Comp(C_{s3};1000)=1
	Pd(3)=1	C_β(3;3)=1	Comp(D_{s3};3)=1
S_4	Pc(1100)=0.25	C_α(1100;1000)=0	Comp(C_{s4};1000)=0.75
	Pc(1000)=0.5	C_α(1000;1000)=1	
	Pc(900)=0.25	C_α(900;1000)=1	

	Pd(3)=0.25	$C_\beta(3;3)=1$	$Comp(D_{s4};3)=0.375$
	Pd(5)=0.25	$C_\beta(5;3)=0$	
	Pd(4)=0.25	$C_\beta(5;3)=0$	
	Pd(3.5)=0.25	$C_\beta(3.5;3)=0.5$	

Table 8: Global cost and delay compatibility for filtered scenarios

Step 3.4 of Figure 36 refers to the extraction of all the scenarios available in the knowledge base in order to propose a first version of the process to the process manager. Moreover, all the information computed during this step is presented to the process manager in the V_0 dashboard. Then, the first version of the process (V_0) and its associated dashboard are illustrated in Figure 38.

After analysing V_0 dashboard, the process manager decides to designate a preferred scenario that users will follow during process execution. The chosen path is scenario S_2. It is only a piece of advice, and this choice can be changed during the execution of the process.

Once V_0 is defined and adjusted (if necessary) by the process manager, the process execution takes place. The process execution is described in the next section.

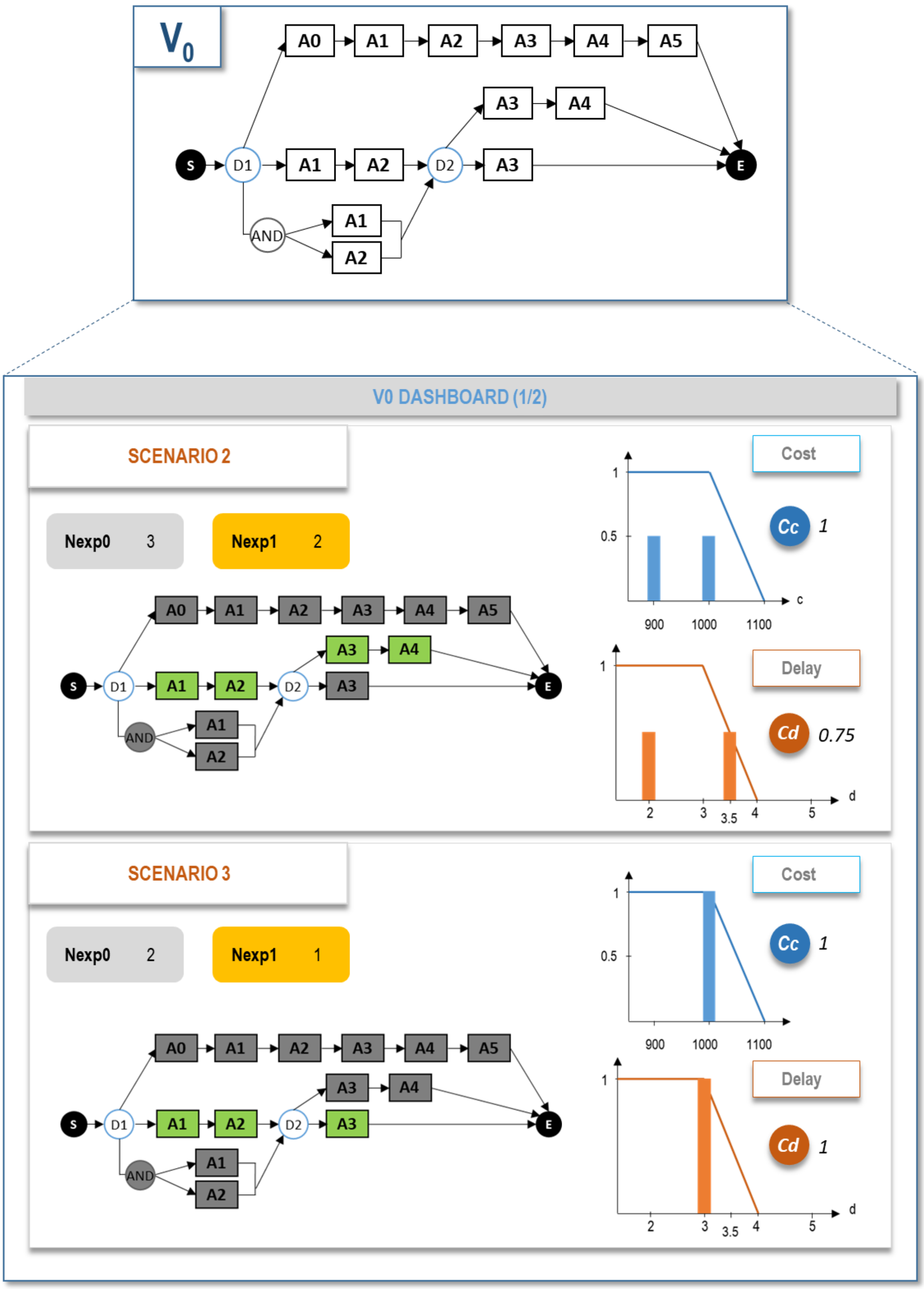


Figure 38: First version of the process (V_0) and its dashboard – Illustration (1/2)

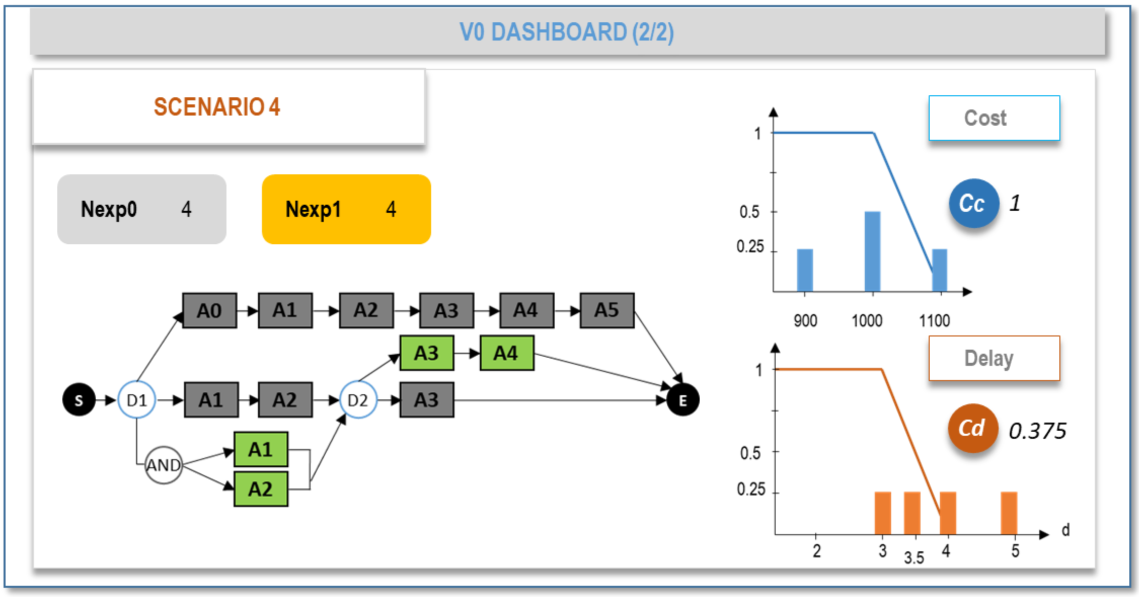


Figure 38 (continuation): First version of the process (V_0) and its dashboard – Illustration (2/2)

4.1.4 Step 4. Process execution. Continuous adaptation

During this step, users carry out the agile process using the first version adapted by the process manager as a guideline. They perform the process following the preferred scenario and at each decision-making point, they decide what to do next. Decision makers can propose and implement options not provided in V_0, and modifications can be inserted in real time. In order to support decision making, a decision dashboard is available at each decision-making point. Similar to the V_0 dashboard, presented in section 4.1.3, the decision dashboard provides information and graphic representations regarding different indicators of filtered scenarios. Moreover, after each decision point, the rationale that led to the decision is formalized by the problem-solving team, and will be stored along with the experience and its indicators in the experience base in the last lifecycle step.

The execution of the activities of the agile process is performed by users. They begin to carry out the process based on the first version provided by the process manager. If the process execution is standard, users should follow exactly the preferred path chosen by the manager during the previous step. When a decision-making point is reached, decision makers choose which scenario to perform, helped with information provided in the decision dashboard (see subsection "decision dashboard" below). If none of the available options is adequate for the process, they can decide to create a new option (i.e. new activities).

During the process execution phase, several versions of the process are created to capitalize the knowledge provided, mainly during decision making. For this purpose, every time that the process is modified, mostly during decision making, a new version is created (see section 3.3). A new version is created at each decision-making point in order to memorize which option was chosen during decision making. Every version of the process includes an explanation that details the reasons for choosing one scenario over another, mainly when the scenario proposed by the system was not chosen. This explanation, called rationale for decisions, is integrated during decision making in a free text form. Every time a new process is defined, the rationale of all past experiences is available in the dashboard to help decision makers.

Moreover, if necessary, users can negotiate with stakeholders if constraints are not satisfied during the execution of the process. Let us consider the selection of one scenario among a set of scenarios during decision making. If the chosen scenario does not satisfy the constraints set by the stakeholders, two options are possible, as illustrated in Figure 39. Either another scenario from the set can be selected, or unsatisfied constraints can be negotiated with their owner. It means that, if cost and/or delay fuzzy constraints are not satisfied during the execution of the process, users can then negotiate their release with the corresponding stakeholder. In the case that the stakeholder determines that the impact of releasing such a constraint is minor, the fuzzy constraint may be changed to its new *allowed* value. If the constraint is not released, another option to perform the process (i.e. another scenario) should be considered.

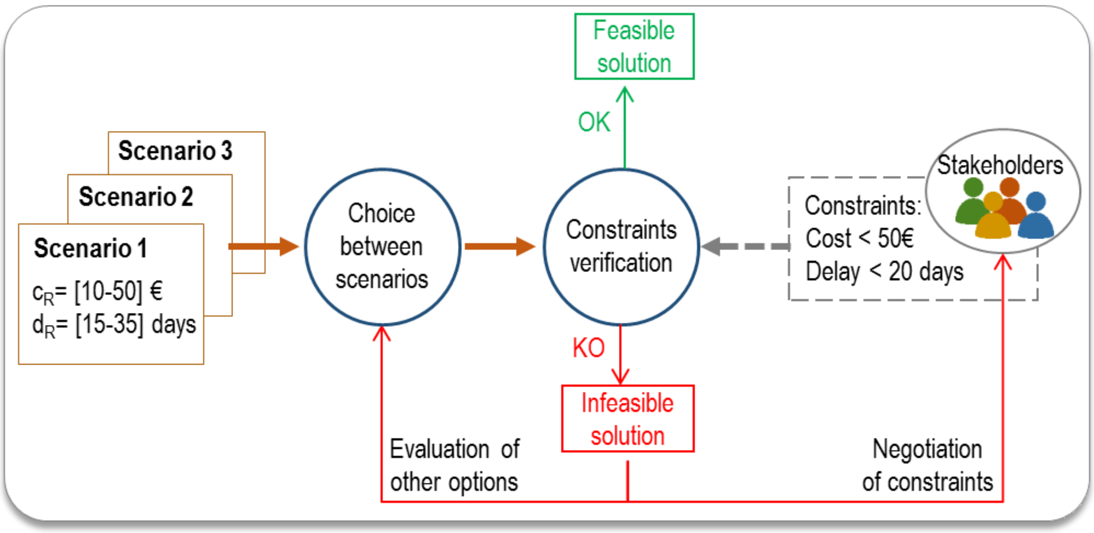


Figure 39: Constraints negotiation

In the next subsection, the decision dashboard is detailed and the manner of computing indicators is described.

Decision dashboard

A *decision dashboard* is proposed to users at each decision-making point in order to support decision making. As it has been described, the purpose of the dashboard is to help decision making by providing indicators and graphic distributions for available scenarios, based on the process execution state. Unlike V_0 dashboard, decision dashboards display information from two perspectives or views. First, the "*action view*" refers to the action that needs to be taken right after the decision (i.e. to choose one alternative or another). It presents computed indicators for all scenarios that share the same action after the decision-making point. Second, the "*scenario view*" presents indicators for each filtered scenario (similar to V_0 dashboard) and helps to evaluate the complete process after the decision-making point. In order to compile information and to build the decision dashboard, first the scenario view is defined followed by the action view.

1) Scenario view

The scenario view displays the available filtered scenarios that could be performed after the decision-making point. For each scenario, relevant information is provided in order to help decision making. Such information are: the number of experiences contained in the scenario before and after filtering (Nexp0 and Nexp1), the probability and compatibility distributions for cost and delay for each scenario, and the global compatibilities values of cost (Cc) and delay (Cd).

The decision dashboard is proposed to users during a given decision, then, in order to compute cost and delay compatibility values, the process progression should be analysed at that precise moment. For each experience activity, its progression state is considered, which can take the values "finished" (the activity is completed) or "unfinished" (the activity is not completed). If an activity is being performed at the time of the decision, it is considered as unfinished.

Moreover, each scenario activity is defined as "selected" or "non-selected" taking into consideration the result of the previous decisions. Thus, all the activities that could be performed respecting the result of the decision (the action that was chosen to execute the process) are considered as "selected". For instance, let us consider the illustration presented in previous sections. Let us consider that during the decision D1 the action a2 is chosen to execute the process, then all the scenario activities linked to that action are considered as Selected (Figure 40). All the scenario activities linked to actions a1 and a3 are set as Non-Selected.

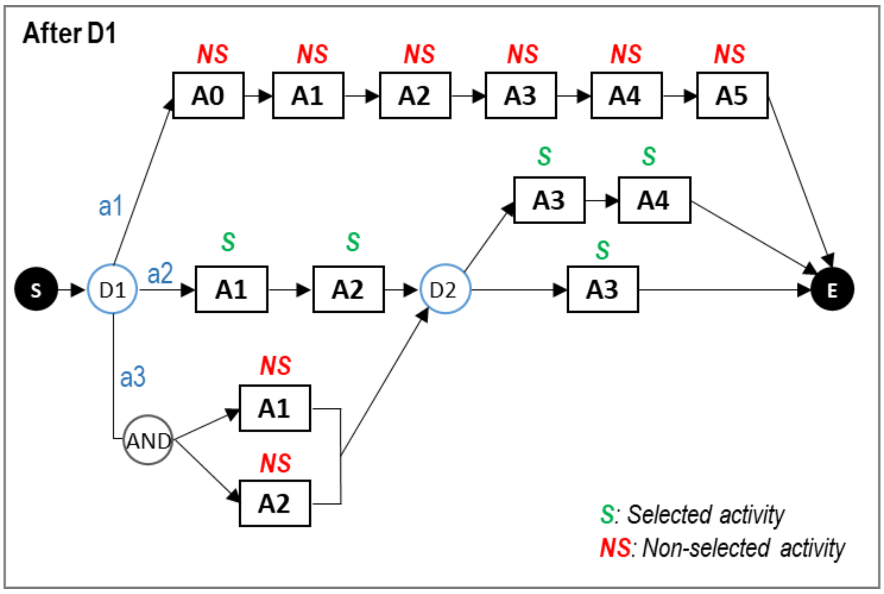


Figure 40: Selected and Non-Selected activities

For each experience of the filtered scenario, at the time of the decision, the global cost value can be computed as the sum of: the cost of the finished activities of the current process (denominated real cost), plus the cost of the unfinished activities from past experiences (denominated cost of the activity of the experiences). The function (denoted fc(V, E, Act)) which allows the computing of the global cost is represented in Algorithm 3. The same is performed for delay global values, the real delay is considered for finished activities, and the activity delay (delay of the activity of the experiences) is considered for unfinished activities. The recursive function (denoted as fd(V, E, Act)) which allows the computing of delay is also described in Algorithm 3.

Moreover, considering such global values, the probabilities of cost and delay values are computed with Algorithm 3. Once probabilities are computed, the compatibility of such cost and delay compared to the soft constraints are given.

```
Declare variables $S_i$, $S_{TP}$, $E_{ij}$, $SE'_i$, $D_i$, $C_i$, $d_{ij}$, $c_{ij}$, $c_k$, $d_k$, Nbck, $Pc_{ki}$,
                  $Nbd_k$, $Pd_{ki}$
Declare functions fd(V, E, Act), fd_finished(V, Act), fc(V, E, Act)
Initialize variables $D_i = 0$, $C_i = 0$, $Nbc_k = 0$, $Pc_{ki} = 0$, $Nbd_k = 0$, $Pd_{ki} = 0$,
delay = 0, $c_{ij} = 0$

Function: fd(V, E, Act)
 If Act = start(E) then
   Return 0
 Else
   max = 0
   For a $\in$ pred(E, Act)
      If a.finished = true and a.selected = true then
        delay = fd(V, E, a)
      else
        delay = fd_finished(V,a)
      End If
      If delay > max then
        max = delay
      End If
   End For
 Enf If
Return max + duration(E, Act)
End

Function: fd_finished(V, Act)
 If Act = start(V) then
   Return 0
 Else
   max = 0
   For a $\in$ pred_finished(V, Act)
     delay = fd_finished(V, a)
     If delay > max then
       max = delay
     End If
   End For
 End If
Return max + real_duration(V, Act)
End

Function: fc(V, E, Act)
  C = 0
 For each a $\in$ E
   If a.finished = true
     C = C + real_cost(V, a)
   Else
     C = C + cost(E, a)
   End If
 End For
Return C
```

__V__: Version of the process
__E__: An experience
__Act__: Activity from which the cost or delay is computed

__pred(E, Act)__: List of predecessors of Act in the experience E
__pred_finished(V, Act)__: List of predecessors of Act in the version V, which have a state=finished
__duration(E, Act)__: duration of the activity Act in the experience E
__real_duration(V, Act)__: Real duration of the activity Act in the version V
__start(E)__: First fictive node of the experience E
__end(E)__: Last fictive node of the experience E
__real_cost(V, a)__: Real cost of the activity a in the version V
__cost(E, a)__: Cost of the activity a in the experience E

```
End
Start
 For all $S_i \in S_{TP}$
   For $E_{ij} \in SE'_i$
     $d_{ij} = fd(V_c, E_{ij}, end(E_{ij}))$
     $D_i = D_i \cup d_{ij}$
     $c_{ij} = fc(V_c, E_{ij}, end(E_{ij}))$
     $C_i = C_i \cup c_{ij}$
   End For
   For each $c_k \in C_i$
     For $E_{ij} \in SE'_i$
       If $(c_k = c_{ij})$ then
         $Nbc_k = Nbc_k + 1$
       End if
     End For
     $Pc_{ki} = \frac{Nbc_k}{|SE'i|}$
   End For
   For each $d_k \in D_i$
     For $E_{ij} \in SE'_i$
       If $(d_k = d_{ij})$ then
         $Nbd_k = Nbd_k + 1$
       End if
     End For
     $Pd_{ki} = \frac{Nbd_k}{|SE'i|}$
   End For
 End For
End
```

S_i: *Scenario i*
S_{TP}: *Set of scenarios corresponding to the selected type of process*
E_{ij}: *Experience j of the scenario i*
SE'_i: *Filtered set of experiences*
V_c: *Current version*
D_i: *Set of different delays of the experiences linked to SE'_i*
C_i: *Set of different costs of the experiences linked to SE'_i*
d_{ij}: *Delay of the experience j of the scenario i*
c_{ij}: *Cost of the experience j of the scenario i*
Nbc_k: *Number of occurrences of c_k*
Nbd_k: *Number of occurrences of d_k*
Pc_{ki}: *Probability that c_k exists within SE'_i*
Pd_{ki}: *Probability that d_k exists within SE'_i*

Algorithm 3: Compute of cost and delay probabilities for decision dashboard – Scenario view

Finally, the global compatibility of cost and delay are computed with equations (5) and (6).

2) Action view

The action view displays the possible actions to take right after the decision. For each action, aggregated information from the corresponding scenarios is presented. Displayed indicators are: the global number of experiences “Nexp1” (i.e. sum of all experiences related to the action); the “flex” indicator that provides the number of available options (i.e. scenarios) to perform the process if that action is chosen; the probability and compatibility distributions for cost and delay for the set of scenarios; and the global compatibility values of cost (Cc) and delay (Cd).

Information from each scenario is aggregated upon its corresponding action. For this purpose, all scenarios for which a given action is performed (i.e. all scenarios that go through a given action) need to be selected in order to compute their global compatibility values. The algorithms and formulas to be used during this step are similar to those used to compute view perspective indicators, since the global cost and delay are given by the sum of the values of the current process activities and the values of each past experience. However, the sets of cost and delay values considered are given by the union between all the values of all the filtered experiences that pass by the action. This means that, considering a given action, for all filtered experiences that go through it, their cost and delay values are taken into account in order to define the cost and delay sets that will allow the computing of probabilities and compatibilities.

For this purpose, Algorithm 4 describes the manner to define the set of cost and delay values in order to compute probabilities. Considering these values, cost and delay compatibilities can be computed.

```
Start
Declare variables $E_{ij}$, $SE'_i$, $S_i$, $S_{lm}$, $D_i$, $C_i$, $d_{ij}$, $c_{ij}$, $D_{lm}$, $C_{lm}$, $c_k$, $d_k$,
                  $Nbck$, $Pc_{klm}$, $Nbd_k$, $Pd_{klm}$

Initialize variables $D_{lm} = \emptyset$, $C_{lm} = \emptyset$, $D_i = \emptyset$, $C_i = \emptyset$, $Nbc_k = 0$, $Pc_{klm} = 0$,
                     $Nbd_k = 0$, $Pd_{klm} = 0$

 For each $S_i \in S_{lm}$ ($S_{lm} \subseteq S_{TP}$)
  For each $E_{ij} \in SE'_i$
   $D_i = D_i \cup fd(V_c, E_{ij}, end(E_{ij}))$
   $C_i = C_i \cup fc(V_c, E_{ij}, end(E_{ij}))$
  End For
  $D_{lm} = D_{lm} \cup D_i$
  $C_{lm} = C_{lm} \cup C_i$
 End For
 For each $c_k \in C_{lm}$
  Nbexp = 0
  For each $S_i \in S_{lm}$
   For each $E_{ij} \in S_i$
     If ($c_k = c_{ij}$) then
       $Nbc_k = Nbc_k + 1$
.....End If
     Nbexp = Nbexp + 1
   End For
  End For
  $Pc_{klm} = \frac{Nbc_k}{Nbexp}$
 End For
 For each $d_k \in D_{lm}$
  Nbexp = 0
  For each $S_i \in S_{lm}$
   For each $E_{ij} \in S_i$
     If ($d_k = d_{ij}$) then
       $Nbd_k = Nbd_k + 1$
.....End If
     Nbexp = Nbexp + 1
   End For
  End For
  $Pd_{klm} = \frac{Nbd_k}{Nbexp}$
 End For
 End
```

S_{lm}: *Set of scenarios concerned by the action m of the decision l*
D_{lm}: *Set of different delays of the scenarios linked to S_{lm}*
C_{lm}: *Set of different costs of the scenarios linked to S_{lm}*
E_{ij}: *Experience j of the scenario i*
SE'_i: *Filtered set of experiences*
D_i: *Set of different delays of the experiences linked to SE'_i*
C_i: *Set of different costs of the experiences linked to SE'_i*
d_{ij}: *Delay of the experience j of the scenario i*
c_{ij}: *Cost of the experience j of the scenario i*
Nbc_k: *Number of occurrences of c_k*
Nbd_k: *Number of occurrences of d_k*
Pc_{klm}: *Probability that c_k exists within S_{lm}*
Pd_{klm}: *Probability that d_k exists within S_{lm}*
Nbexp: *Total number of experiences corresponding to the action lm*

Algorithm 4: Computing of cost and delay probabilities for decision dashboard – Action view

Therefore, the global compatibility of cost and delay corresponding to an action lm are computed using:

$$Comp(C_{lm}, TC) = \sum_{c_k \in C_{lm}} Pc_{klm} * Compat_{\alpha}(c_k, TC) \qquad (7)$$

$$Comp(D_{lm}, TD) = \sum_{d_k \in D_{lm}} Pd_{klm} * Compat_{\beta}(d_k, TD) \qquad (8)$$

4.1.4.1 Illustrative example

Let us consider the execution of the agile problem-solving process illustrated in the previous sections. Users execute the process following the preferred scenario proposed by the process manager. Then, scenario S_2 should be chosen over the other scenarios. Let us consider the evolution of the process from V_0 to V_1 as illustrated in Figure 41.

The current version of the process is V_1, it has been created after decision D1, and the next decision-making point is D2. In order to construct the decision dashboard, cost and delay need to be updated with the values from the already performed activities (A1 and A2). The real duration of activities A1 and A2, and the duration of A3 and A4 for experiences E_{21}, E_{22} and E_{31} are shown in Table 9. The total cost and delay values are presented in the lower rows.

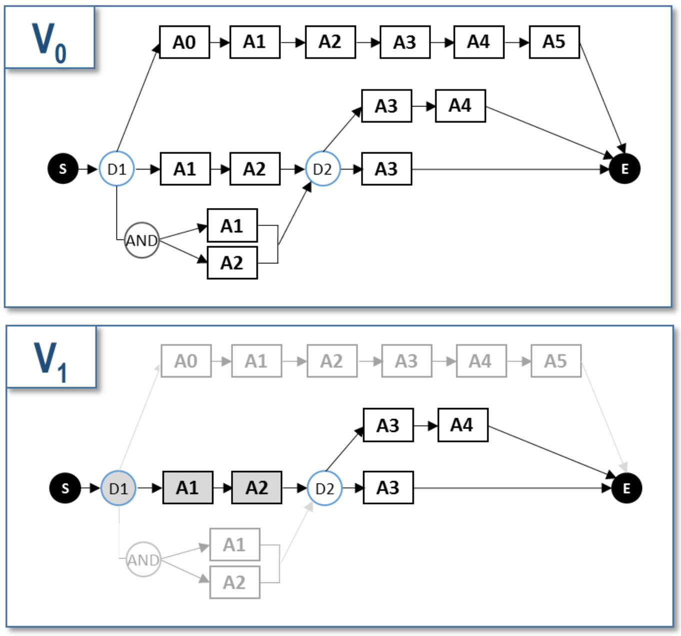


Figure 41: Versions V_0 and V_1 of the process - Illustration

Activity	Real		E_{21}		E_{22}		E_{31}	
	Cost (€)	Delay (days)	Cost (€)	Delay (days)	Cost (€)	Delay (days)	Cost (€)	Delay (days)
A1	300	0.5	-	-	-	-	-	-
A2	400	1	-	-	-	-	-	-

A3	-	-	100	1	200	0.5	300	2
A4	-	-	100	1	100	1	-	-
Total		Cost	900		1000		1000	
		Delay	3.5		3		3.5	

Table 9: Activities cost and delay values - Illustration

With the new values of cost and delay for each experience, probability, compatibility and global compatibility can be computed for each scenario as presented in Table 10.

Scenario	Probability	Compatibility	Global compatibility
S_2	Pc(900)=0.5	C_{α}(900;1000)=1	Comp (C_{s2};1000)=1
	Pc(1000)=0.5	C_{α}(1000;1000)=1	
	Pd(3.5)=0.5	C_{β}(3.5;3)=0.5	Comp (D_{s2};3)=0.75
	Pd(3)=0.5	C_{β}(3;3)=1	
S_3	Pc(1000)=1	C_{α}(1000;1000)=1	Comp (C_{s3};1000)=1
	Pd(3.5)=1	C_{β}(3.5;3)=0.5	Comp (D_{s3};3)=0.5

Table 10: Probability and compatibility values at decision D2 – Illustration

In order to build the decision dashboard, the probability distributions and the compatibilities are considered. First, the scenario view is constructed from values of Table 10. Second, for the action view, the same values of the scenarios are considered, since in this example for each action there is only one scenario. If several scenarios were available, the set of their cost and delay values would have been considered in order to define the action probability distributions and compatibility. The decision dashboard is presented in Figure 42.

The “action views” are local to the decision and, on the other hand, the “scenario views” are global to the process.

Let us consider that, based on the decision dashboard, decision makers choose to perform scenario S_2. The rationale that led decision makers to choose scenario 2 should be described at this point, for simplicity reasons it is not illustrated in this example. Then, activities A3 and A4 are normally performed and recorded in the third version of the process (V_2) as illustrated in Figure 43.

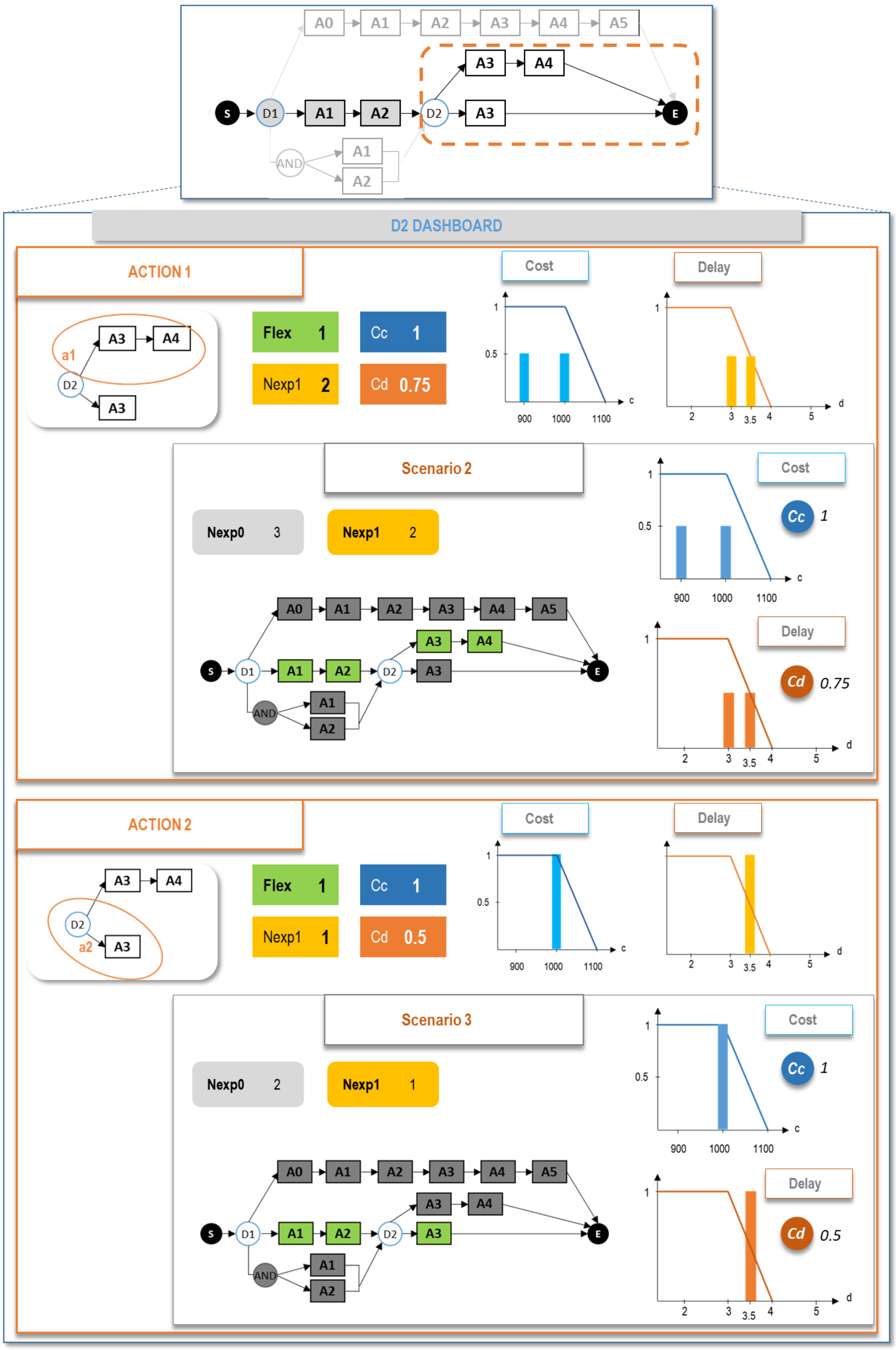


Figure 42: D2 dashboard - Illustration

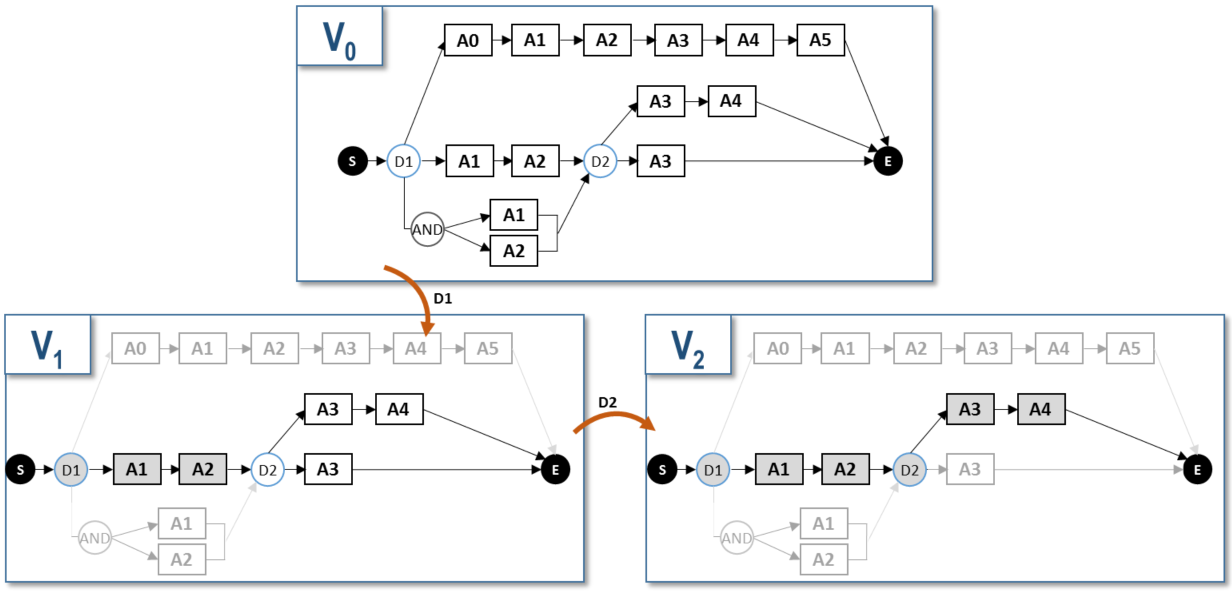


Figure 43: Versions V_0, V_1 and V_2 of the process - Illustration

Once process execution is over, the last version of the process is considered to be stored in the experience and/or knowledge base. The process storage is described in the next section.

4.1.5 Step 5. Storage in the Experience/Knowledge Base

When the agile process is over, all the information that helps to describe and characterize the process needs to be properly stored in the experience and knowledge bases. The storage phase is of major importance since it guarantees the knowledge and experience capitalization (and reuse). This step is performed by users, except in the case where a new scenario needs to be added in the knowledge base.

During this step, the last version of the agile process, created on step 4, is analysed in order to determine whether the process structure has changed or not. Two options are possible. First, when there is no change in the structure of the process, it can be stored linked to the scenario that represents its structure in the experience base. It means that the agile process has now become a new experience corresponding to that scenario. This option also includes the possibility that the structure of the process has changed but it corresponds to another existing scenario. Then, it is also stored in the experience base as a new experience linked to that other scenario.

On the other hand, when a change in the process structure is observed and its structure does not match any existing scenario, a knowledge expert needs to be involved. The knowledge expert must decide if the new scenario is validated or not. In the case that it is validated, it is added as a new scenario in the knowledge and experience bases. Moreover, in the experience base, the execution of the process is stored as the first experience corresponding to the scenario.

4.1.5.1 Illustrative example

In order to store the current process, the knowledge expert verifies if the structure of the process has changed. Considering the last version of the process illustrated in Figure 43, the structure of the process (A1-A2-A3-A4) corresponds to an existing scenario (S_2). Then, the process is stored into the experience base as a new experience.

It means that the experience base is updated with the new experience. Table 11 shows the available experiences in the experience base for the type of process: "Problem solving" (Table 5 has been updated and is represented in Table 11).

Scenario	Experience	T1	T2	T3	T4
S_1	E_{11}	Complex	Quality	Saddle	Painting
S_2	E_{21}	Simple	Quality	Frame	Dimensional
	E_{22}	Simple	Manufacturing	Pedals	Painting
	E_{23}	Simple	Finance	Saddle	Material
	E_{24}	Simple	Quality	Pedals	Painting
S_3	E_{31}	Simple	Quality	Front	Painting
	E_{32}	Simple	Manufacturing	Saddle	Assembling
S_4	E_{41}	Simple	Quality	Frame	Painting
	E_{42}	Simple	Quality	Wheels	Painting

	E_{43}	Simple	Manufacturing	Pedals	Painting
	E_{44}	Simple	Manufacturing	Pedals	Material

Table 11: Updated experiences tags – Illustration

In the last sections, the lifecycle that allows the process manager and users to define and to execute an agile problem-solving process has been introduced. Principles regarding decision making based on event are detailed in the next section.

4.2 Decision making based on event

Two types of decisions can be found in an agile process. Nominal decision-making principles were introduced in section 3.2. On the other hand, when decision-making points are created in order to face an unexpected change, they are denominated decision-making points "based on event" in our study. If an unexpected event arises during the execution of the process, a decision-making point based on event is created in order to analyse, decide and find a solution. The path chosen to adapt the process to face the unexpected event will be defined and stored in the experience and knowledge bases. Then, if the event reoccurs in the future, all existing scenarios linked to such event will be proposed to users as a possible solution.

Decisions based on event introduce a change in the model since they need a special treatment for their storage and future reuse. However, the basic elements of the model stay unchanged. Unexpected events can appear during the execution of the process, then, decision makers need to decide what to do next. The event may need to be immediately treated, in which case a new decision-making point based on event needs to be created in order to decide which path to follow. Afterwards, it can be chosen to create a new scenario (new activities or new sequence of the same activities) or to perform the same existing activities to overcome the problem. In addition, if an activity is being performed at the time that the unexpected event occurred, it is considered as an "unfinished" activity. Then, during the newly created decision based on event, it can be chosen to create activities and to resume the "unfinished" activity, or to perform other activity. When the process is over, it is stored into the experience and the knowledge base.

For simplification purposes, when the first version of the process is proposed, only "nominal" (i.e. not based on event) scenarios are visible (V_0). However, when an unexpected-event occurs during the process execution, all the activities or scenarios that were performed in the past to overcome such an event are "activated". It means that if activities are not concerned by such an event, they are inhibited and they are not available during the process execution. As briefly mentioned in section 3.5.1, nodes have an attribute named "list of events" that refers to all the events that can "activate" such a node, as presented on the UML diagram of Figure 19. The list can take the values *None, $Event_1$*, *$Event_2$*, etc. When no unexpected event has occurred, the events list for all nodes should be: *None*. It means that, if a scenario has only been performed in a nominal mode, the attribute value "*None*" indicates that when an unexpected event occurs, it should be inhibited (i.e. the activities that compose it). If on the other hand, a scenario was created to overcome an unexpected event *$Event_1$*, "*$Event_1$*" should be added to the "list of events" attribute of the concerned activities, then, it should only be activated if the event named *$Event_1$* reoccurs.

The storage of the process is carried out like a regular experience. If the structure of the process corresponds to an existing scenario, it is stored as a new experience. If the process structure has changed, a new scenario needs to be created. In both cases, for the activities (or other nodes) that were performed after the unexpected event, their list of events needs to be properly modified. The event that activates such nodes needs to be added to the list.

4.2.1 Illustrative example

The same example from previous sections is considered. Let us consider that we are at the moment of the execution of the activity A4 (defining preventive actions). At that moment, an unexpected event occurs, as illustrated in Figure 44. The unexpected event is implied by the fact that the problem reappeared.

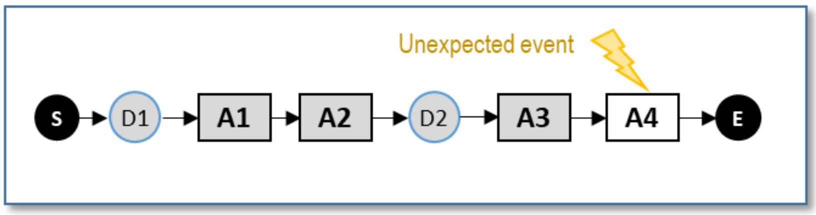


Figure 44: Occurrence of an unexpected event - Illustration

Such an event had never occurred during problem solving within the company. It means that when the unexpected event appeared, no scenario based on event could be activated based on experience feedback. A decision-making point based on event is created in order to immediately treat the event (D3). Decision makers decide to go back to activity A2 in order to redefine the root cause of the problem. Afterwards, activities A3 and A4 need to be performed again. This decision creates a new version of the process, presented in Figure 45.

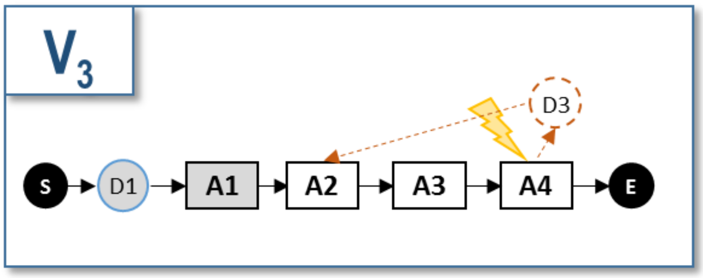


Figure 45: Version V_3 of the process - Illustration

Let us now consider that the process is completely carried out without further problems. At the moment of the storage of the last version of the process, the attributes of the activities need to be updated including the event "*Event$_1$*" in the event list attribute. Then, if the unexpected event "*Event$_1$*" reoccurs in the future, activity A2 (and followed by A3 and A4) will be activated as a possible solution. The decision-making point based on event "D3" is stored into the knowledge base as a XOR node (with the event "*Event$_1$*" in the event list attribute).

4.3 Conclusion

The lifecycle of an agile problem-solving process was defined in this chapter. Such a lifecycle, based on CBR principles, allows to define and to execute an agile problem-solving process. The importance of knowledge and experience capitalization and reuse principles along the agile lifecycle has been outlined.

First, the agile problem-solving process needs to be characterized. For this purpose, a set of tags is defined by the process manager, the person in charge of the definition and the execution of the agile lifecycle, and constrained with cost and delay limits defined by stakeholders. The defined set of tags allows to filter the experience base in order to retrieve past experiences with similar characteristics to the current process. A set of indicators regarding filtered experiences is proposed to the process manager in a V_0 dashboard, which, along with the set of scenarios in the knowledge base will constitute the process first version. The first version of the process is proposed to users as a guideline to be used during process execution. During the process execution, at decision-making points, decision dashboards compile the information regarding filtered scenarios in order to support decision making. Moreover, the rationale of each decision is stored in order to allow its consultation during future processes. Finally, the agile process is stored into the experience base. In the case that the structure of the process does not match any existing scenario, and once validated by the knowledge expert, a new scenario is created in the experience and knowledge bases.

The treatment of unexpected events was described at the end of this chapter. Such an extension of the model allows the process to be agile in order to respond to events that were not considered when defining nominal scenarios for executing the process. Unexpected events create decision-making points based on event, which allow decision makers to define how to treat such an event. The process lifecycle is the same for decisions based on event, except for the storage step which needs special treatment. When an unexpected event has been treated (the process has been updated to face this problem), the solution is memorized within the corresponding experience in order to be reused if the same kind of event reoccurs.

5. ILLUSTRATION BASED ON AN INDUSTRIAL SITUATION

In the previous chapters the agile problem-solving process model has been detailed. In order to validate and to improve such a model, its application to the real case of a company is described in this section.

The case study was conducted at a surface treatment company dedicated to the aeronautical sector. The company performs activities such as surface treatment, painting, non-destructive testing and chemical milling. The company Axsens-bte and the laboratory LGP-ENIT have come to work together with this surface treatment company in the framework of a collaborative project.

The work conducted in the company consisted mainly on two steps, first the interview of the quality department employees about different problems that they had needed to overcome. Second, ProWhy, a computer-based platform used to trace and to solve problems, was consulted to analyse the problems solved and traced within it. Both inputs were considered to simulate a knowledge and experience base for the company, and afterwards, to recreate the agile lifecycle for solving a problem. In order to describe the case study, knowledge and experience bases are presented in section 5.1, and the agile lifecycle is introduced from section 5.2 to section 5.6.

5.1 Knowledge and experience base

Interviews to the quality department employees (see interview template in Appendix 2) were conducted with two main purposes. First, the determination of the problem-solving approach (or approaches) deployed within the company. Second, the collection of real problem-solving cases in order to build the experience and knowledge bases. Two types of problem-solving methods were identified. First, the method 8D is used within the company in three situations: 1) when the resolution of the problem is demanded and/or its treatment is followed by customers, 2) when the problem is complex, and/or 3) when the problem has reoccurred several times. Second, the method to treat all other problems is called Non-Conformity treatment (NC). Moreover, all 8D-problems are stored in the software ProWhy, designed to structure, store and share information about problem solving.

The knowledge and the experience bases were created from collected information. The knowledge and the experience bases, corresponding to the type of process problem solving are illustrated in Figure 46. Activities performed to solve problems are: A1: Start immediate containment, A2: Build the team, A3: Define problem, A4: Complete and optimize containment actions, A5: Identify root causes, A6: Define and select permanent corrective actions, A7: Implement permanent corrective actions and check effectiveness, A8: Standardize and transfer the knowledge, A9: Recognize the team and close, and A10: Define and apply immediate preventive actions.

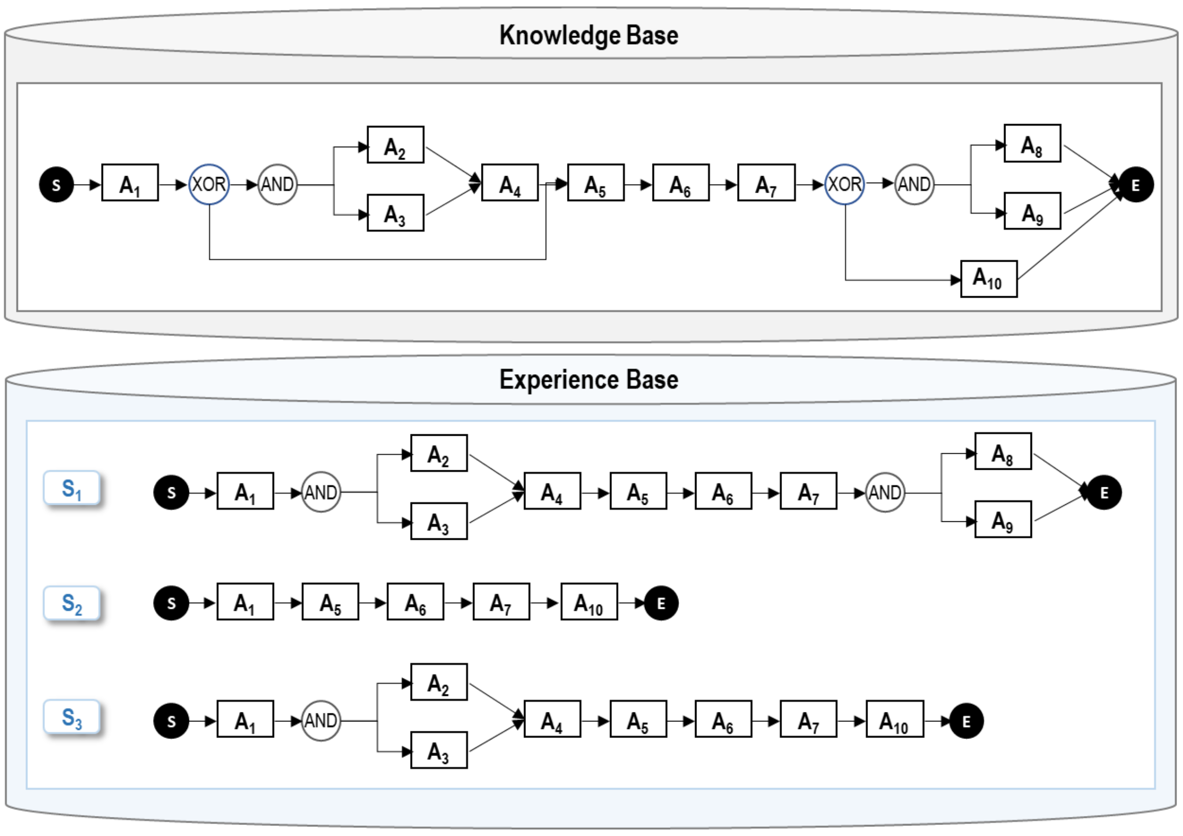


Figure 46: Knowledge and experience base for the type of process: Problem solving

Scenario S_1 of the experience base corresponds to the execution of an 8D problem-solving process. S_2 corresponds to the non-conformity method. Finally, S_3 corresponds to a combination of both methods (i.e. the 8D method is followed until activity A7, but instead of performing A8 and A9, A10 corresponding to the NC method is performed).

Then, each problem collected from the company is treated as an experience and tags are defined in order to characterize each process. Table 12 shows all the experiences contained in the experience base for the type of process "problem solving" including their tags. For the experiences corresponding to S_3, it has been decided that T1=8D since the process is considered as a simplified 8D.

For instance, E_{12} refers to a problem detected by the customer. The reference printed into a big panel was not the correct reference as per specifications. The customer demanded the problem to be solved with an 8D process. The following steps were performed in order to solve the problem:

- Activity A1: In order to immediate contain the problem, all the suspected panels in progress were isolated to control that the printed reference corresponded to the actual part reference.
- Activity A2: A multifunctional team was defined to treat the problem.

- Activity A3: The problem was described. The panel was painted with an incorrect tooling. Both tooling are similar and have similar references.
- Activity A4: The incorrect tooling used to paint the part was safely stored to avoid people using it.
- Activity A5: The identified and validated root causes are: The tooling used had no “poka-yoke” system (i.e. it could be incorrectly used). Operators did not verify the reference of the tooling (i.e. it could be easily confused with other similar tooling).
- Activity A6: Two actions were identified: Incorporating a “poka-yoke” system to the tooling, and reminding the instructions to all operators.
- Activity A7: The verification of the corrective actions was not performed. However, the problem did not reappear.
- Activity A8: In order to standardize the knowledge acquired beyond the scope of this single problem, “poka-yoke” systems were added to all reference-painting tooling.
- Activity A9: The team was thanked for their work and the 8D process was formally closed.

Another example is E_{21}. A dimensional problem was detected in a big part, at the control area of the company. The chemical treatment was not correctly performed, and it resulted in a difference of the thickness of the part. It was decided to treat this problem with the NC method. The following steps were performed:

- Activity A1: One immediate action was defined: all the parts in progress were selected to be controlled.
- Activity A5: The root cause of the problem was identified and validated: The points of electrical contact were modified.
- Activity A6: The corrective actions are: 1) Modification of the assembly line. 2) Perform a chemical analysis of all the parts that present a difference on their thickness with regards to the requirement (identified during A1).
- Activity A7: The actions were validated.
- Activity A10: A preventive action was defined and immediately applied: All of the acceptance criteria were displayed on each work station (i.e. points of electrical contact for each part).

Scenario	Experience	T1	T2	T3	T4
S_1	E_{11}	8D	Quality	Big panel	Part falling
	E_{12}	8D	Quality	Small panel	Tooling
	E_{13}	8D	Quality	Big panel	Dimensional
	E_{14}	8D	Quality	Big panel	Painting
	E_{15}	8D	Quality	Small part	Electrical crash
	E_{16}	8D	Quality	Big part	Dimensional
	E_{17}	8D	Quality	Chemical milling line	Line stoppage
S_2	E_{21}	NC	Quality	Big part	Dimensional
	E_{22}	NC	Quality	Small part	Painting
	E_{23}	NC	Quality	Small part	Dimensional

	E_{24}	NC	Quality	Small panel	Part falling
S_3	E_{31}	8D	Quality	Small part	Dimensional
	E_{32}	8D	Quality	Big part	Painting
	E_{33}	8D	Quality	Small panel	Part falling

Table 12: Experiences tags

Taxonomies for the set of tags are illustrated in Figure 47. The taxonomy for process models (used for tag T1) is presented in Figure 47a, in this case, there are two types of problem-solving processes (8D and NC). The taxonomy for enterprise departments (used for tag T2) is shown in Figure 47b, in this study only the Quality department is considered. The taxonomy for the type of product (used for tag T3) is illustrated in Figure 47c. As several parts are painted and chemically treated in this company, for simplification reasons the types of products are classified into panels (big metallic parts) and parts (all the other parts), the third category is the chemical milling line. Finally, the taxonomy for the types of problems (used for tag T4) is presented in Figure 47d. The problem may have occurred to a part or it can be related to its environment.

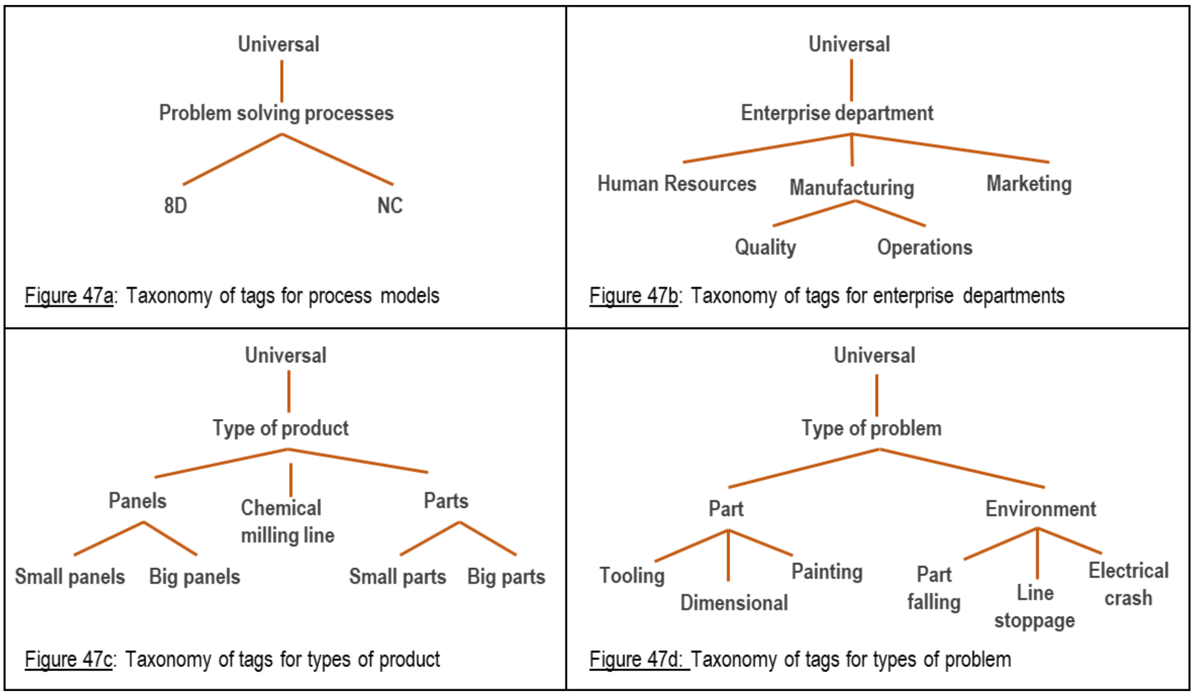


Figure 47: Tags taxonomy

An insight was given on the problem-solving processes stored into the knowledge and experience base of the company. In the next sections a new problem is treated following the agile lifecycle introduced in chapter 4.

5.2 Step 1. Definition of the process scope

A new problem occurs within the surface treatment company. The problem is that a part, ready to be sent to the customer, has fallen from its metallic support. After examination by the final control employee, it has been decided to send the part to the customer since no damage has been noticed. However, the root cause of the part falling needs to be determined in order to avoid its recurrence. The process manager designated for this agile process is the quality manager.

First, the quality manager needs to characterize the process. For this purpose, the table presented in Figure 46 is completed. The method used will be *a priori* the "NC" method (tag T1), since the problem does not seem complex, it has not impacted the client, and it is the first time it appears on this part. The problem will be solved by the quality department (tag T2). The type of product is a big part (tag T3) and the type of problem is part falling (tag T4). As there is no major requirement for either of the tags, the manager decides to set all tags as flexible in order to retrieve several similar past cases. The global similarity threshold is set at 0.8.

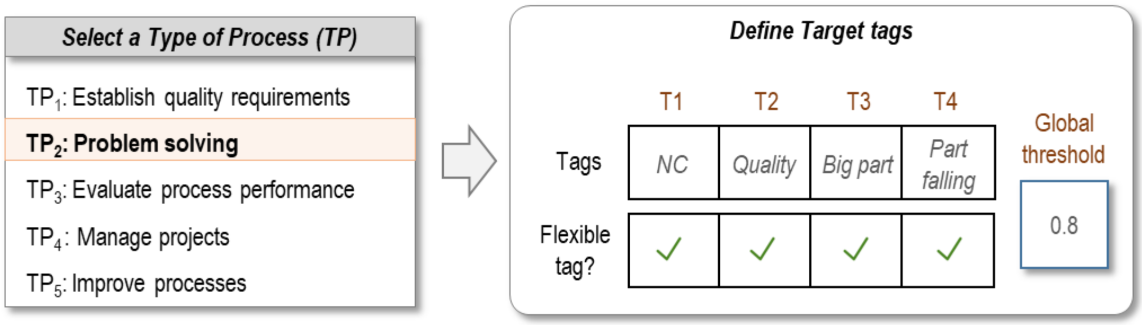


Figure 48: Definition of the process context

Moreover, constraints need to be defined during this step. The quality and the manufacturing manager set the constraints: the cost should not be more than 600€ and the maximum delay is of 8 hours. However, since cost and delay are flexible constraints, the process (quality) manager defines the level of acceptance as: for cost α=0.33, and for delay β =1, as illustrated in Figure 49.

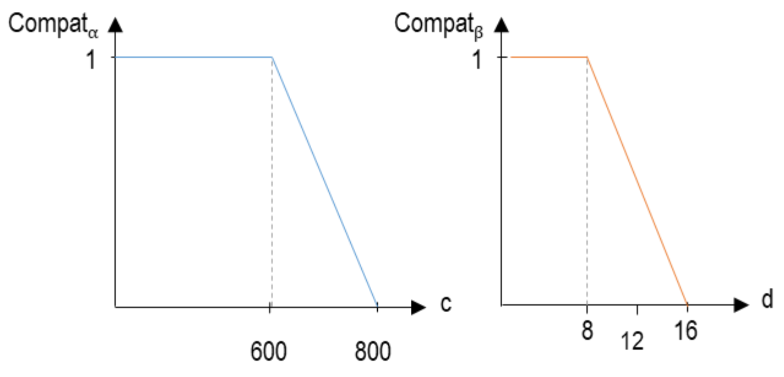


Figure 49: Cost and delay constraints

Once that the process is characterized and the constraints are set, the experience filtering step, described in the next section, is performed.

5.3 Step 2. Experience filtering

In order to filter the experience base, the current process tags are considered in order to compute the similarity between them and the tags of each experience. For this purpose, the similarity formula of Wu and Palmer (see equation (3)) is used.

Table 13 shows the similarity values between the tags of each experience and the target tags (defined in the second row of the table). Once that similarity has been computed, global similarity is computed for each experience using the GOWA operator (see equation (4)). Global similarity is presented in the sixth column of Table 13.

Scenario	**Experience**	**T1**	**T2**	**T3**	**T4**	**Global similarity Sim_g**
TT		**NC**	**Quality**	**Big part**	**Part falling**	
Flexible/Non-flexible?		**F**	**F**	**F**	**F**	
S_1	E_{11}	8D	Quality	Big panel	Part falling	0.8
		Sim=0.67	*Sim=1*	*Sim=0.5*	*Sim=1*	
	E_{12}	8D	Quality	Small panel	Tooling	0.7
		Sim=0.67	*Sim=1*	*Sim=0.5*	*Sim=0.5*	
	E_{13}	8D	Quality	Big panel	Dimensional	0.7
		Sim=0.67	*Sim=1*	*Sim=0.5*	*Sim=0.5*	
	E_{14}	8D	Quality	Big panel	Painting	0.7
		Sim=0.67	*Sim=1*	*Sim=0.5*	*Sim=0.5*	
	E_{15}	8D	Quality	Small part	Electrical crash	0.8
		Sim=0.67	*Sim=1*	*Sim=0.75*	*Sim=0.75*	
	E_{16}	8D	Quality	Big part	Dimensional	0.8
		Sim=0.67	*Sim=1*	*Sim=1*	*Sim=0.5*	
	E_{17}	8D	Quality	Chemical milling line	Line stoppage	0.8
		Sim=0.67	*Sim=1*	*Sim=0.57*	*Sim=0.75*	
S_2	E_{21}	NC	Quality	Big part	Dimensional	0.9
		Sim=1	*Sim=1*	*Sim=1*	*Sim=0.5*	
	E_{22}	NC	Quality	Small part	Painting	0.8
		Sim=1	*Sim=1*	*Sim=0.75*	*Sim=0.5*	
	E_{23}	NC	Quality	Small part	Dimensional	0.8
		Sim=1	*Sim=1*	*Sim=0.75*	*Sim=0.5*	
	E_{24}	NC	Quality	Small panel	Part falling	0.9
		Sim=1	*Sim=1*	*Sim=0.5*	*Sim=1*	
S_3	E_{31}	8D	Quality	Big panel	Dimensional	0.7
		Sim=0.67	*Sim=1*	*Sim=0.5*	*Sim=0.5*	
	E_{32}	8D	Quality	Big part	Painting	0.8
		Sim=0.67	*Sim=1*	*Sim=1*	*Sim=0.5*	
	E_{33}	8D	Quality	Small panel	Part falling	0.8
		Sim=0.67	*Sim=1*	*Sim=0.5*	*Sim=1*	

Table 13: Compute of similarity

In order to perform the filtering of experiences, since there are not any non-flexible tags, the global similarity of each experience is considered. All experiences that have a global similarity inferior to 0.8 are discarded. Then, experiences E_{12}, E_{13}, E_{14} and E_{31} are discarded. Then, after removing experiences E_{12}, E_{13}, E_{14} and E_{31}, the filtered scenarios corresponding to this agile process are obtained (Figure 50).

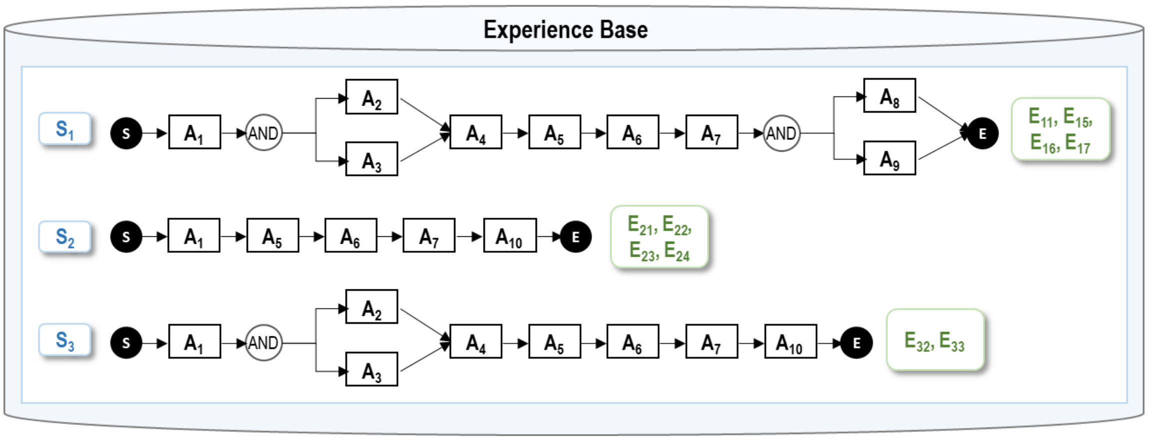


Figure 50: "Filtered" experience base

Once that the filtering of experiences is performed, the first version of the process needs to be defined and adapted. The adaptation of the process first version is detailed in the next section.

5.4 Step 3. Adaptation of the first version of the process

During this step, the first version of the process is adapted by the process manager. For this purpose, the available scenarios from the knowledge base for the type of process "Problem solving" are considered, and the V_0 dashboard is created in order to help the process manager to adapt the first version of the process.

First, the cost and delay values for the filtered experiences are considered (Table 14).

Scenario	Experience	c (€)	d (hours)
S_1	E_{11}	1000	16
	E_{15}	500	16
	E_{16}	600	12
	E_{17}	2000	32
S_2	E_{21}	600	8
	E_{22}	400	12
	E_{23}	800	16
	E_{24}	600	12
S_3	E_{32}	500	8
	E_{33}	400	12

Table 14: Cost and delay values for filtered experiences

In order to create the dashboard, the cost and delay probabilities for the filtered scenarios need to be computed. The "Probability" column of Table 15 presents the probabilities of each value of cost and delay for each scenario. Moreover, the compatibility between such cost and delay values and the constraints needs to be computed with equations (1) and (2). Compatibility values are presented in the "Compatibility" column of Table 15. Finally, the global cost and delay compatibility for each scenario is computed as the sum of the product of probabilities and compatibilities for cost and delay, as presented in the "Global compatibility" column.

Scenario	Probability	Compatibility	Global compatibility
S_1	Pc(1000)=0.25	C_α(1000;600)=0	Comp (C_{s1};600)=0.5
	Pc(500)=0.25	C_α(500;600)=1	
	Pc(600)=0.25	C_α(600;600)=1	
	Pc(2000)=0.25	C_α(2000;600)=0	
	Pd(16)=0.5	C_β(16;8)=0	Comp (D_{s1};8)=0.125
	Pd(12)=0.25	C_β(12;8)=0.5	
	Pd(32)=0.25	C_β(32;8)=0	
S_2	Pc(600)=0.5	C_α(600;600)=1	Comp (C_{s2};600)=0.75
	Pc(400)=0.25	C_α(400;600)=1	
	Pc(800)=0.25	C_α(800;600)=0	
	Pd(8)=0.25	C_β(8;8)=1	Comp (D_{s2};8)=0.5
	Pd(12)=0.5	C_β(12;8)=0.5	
	Pd(16)=0.25	C_β(16;8)=0	
S_3	Pc(500)=0.5	C_α(500;600)=1	Comp (C_{s3};600)=1
	Pc(400)=0.5	C_α(400;600)=1	
	Pd(8)=0.5	C_β(8;8)=1	Comp (D_{s3};8)=0.75
	Pd(12)=0.5	C_β(12;8)=0.5	

Table 15: Cost and delay probability and compatibility values – V0 dashboard

Then, considering such values and the available scenarios from the knowledge base, the V_0 dashboard can be constructed. The V_0 dashboard for this agile process is introduced in Figure 51.

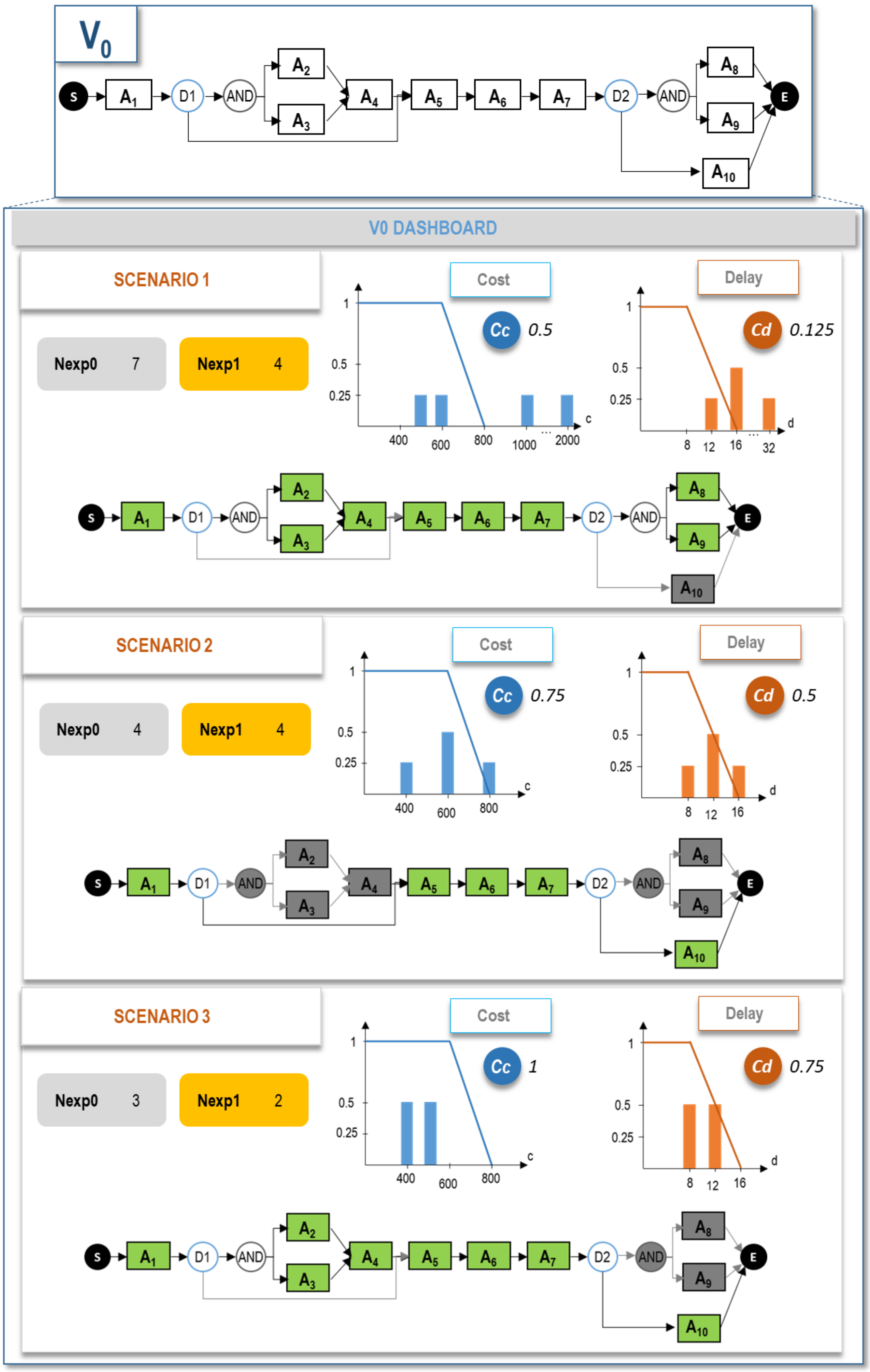


Figure 51: V_0 dashboard

The process manager uses the V_0 dashboard to adapt the first version of the process. Even if S_3 presents good compatibility values, the manager decides not to consider such indicators as the best ones since the scenario does not correspond to the process model “NC” which was defined as the target tag. On the other hand, S_2 is considered more adapted to this problem since cost and delay indicators match the constraint values well. Also, all available experiences for S_2 match the target tags (since Nexp0 is equal to Nexp1, then no experience was discarded). This result conforms to the characterization of the process, since the manager had defined the process model as “NC”, which corresponds to S_2. Also, this result seems correct since an 8D process, used to treat more complex problems, is usually longer and more expensive than a NC process. Then, the process manager defines S_2 as the preferred (but not mandatory) path to perform the process.

When the first version of the process is defined, the process can be executed. The process execution phase is described in the next section.

5.5 Step 4. Process execution. Continuous adaptation

Users will execute the process using the first version defined by the process manager as a baseline. The execution of the process is described in this section. Let us consider that one of the quality department employees (user) will be in charge of this problem-solving process. This user takes into consideration the first version of the process defined by the process manager, which includes a preferred scenario for its execution, and begins to execute the process. When decision-making points D1 and D2 are reached, the decision dashboard needs to be constructed. However, in order to ensure understanding of this step, two different cases are proposed and developed during the agile lifecycle steps 4 and 5 (see Figure 26, chapter 4). The first case regards a normal execution of the process, and the second case considers the occurrence of an unexpected event.

5.5.1 Case 1- Normal execution of the process

First, the problem needs to be contained (activity A1). For this purpose, the first containment action is to perform a deep control the fallen part to verify that it is not damaged. The second action is to control the zone where the incident was produced, in order to ensure that there is no evident cause that could be treated to avoid other parts falling. Finally, once that no damages were found on the part, it is sent to the customer in order to avoid a delay on delivery.

After performing A1, a decision-making point is reached, as illustrated in Figure 52. The user needs to decide which action to choose helped by the decision dashboard. Indicators must be computed for the decision dashboard. For this purpose, the cost and delay of each activity must be considered. Since A1 has already been performed (i.e. the activity state is “finished”), its real cost and delay values are considered.

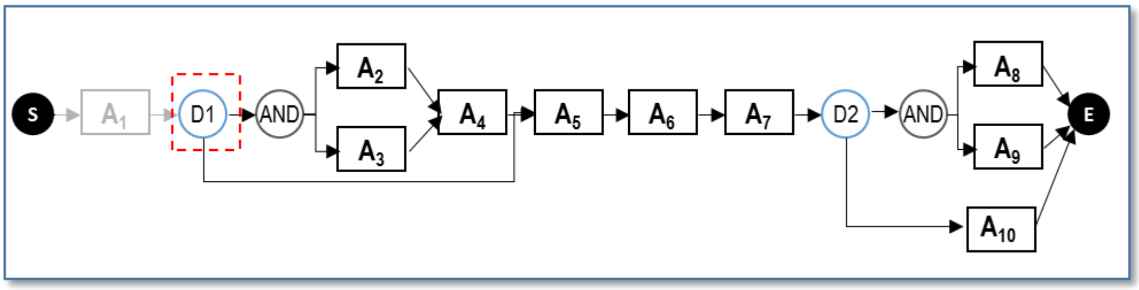


Figure 52: Evolution of the agile process

Table 16 shows the cost and delay of every activity for each filtered experience. The lowest rows of Table 16 indicate the global values of cost and delay for each experience, computed considering the real value of the current activity A1 plus the remaining values of the activities from each experience.

Considering the updated cost and delay values for each experience, probabilities and compatibilities can be computed. Cost and delay probabilities, compatibilities and global compatibilities are presented in Table 17.

Indicators must also be computed for the action view of the dashboard. For this purpose, all scenarios that go through each action need to be identified. For action a1 (the action that is followed by activities A1 and A2 performed in parallel), the scenarios that go through it are: S_1 and S_3. For the action a2 (the action that is followed by activity A5) only one scenario is identified: S_2. In order to build the dashboard, all cost and delay values for each action are defined as the union of all the values corresponding to each scenario that pass through the action. Then, the probability and compatibility values can be computed as Table 18 indicates.

Considering the computed probability and compatibility values, the dashboard for decision D1 is built (Figure 53).

Activity	Real		E_{11}		E_{15}		E_{16}		E_{17}		E_{21}		E_{22}		E_{23}		E_{24}		E_{32}		E_{33}	
	Cost (€)	Delay (hour)	Cost (€)	Delay (hour)	Cost (€)	Delay (hour)	Cost (€)	Delay (hour)	Cost (€)	Delay (hour)	Cost (€)	Delay (hour)	Cost (€)	Delay (hour)	Cost (€)	Delay (hour)	Cost (€)	Delay (hour)	Cost (€)	Delay (hour)	Cost (€)	Delay (hour)
A1	100	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
A2	-	-	100	2	50	2	50	1	100	2	-	-	-	-	-	-	-	-	50	1	50	2
A3	-	-	100	2	50	2	50	2	200	4	-	-	-	-	-	-	-	-	50	1	50	1
A4	-	-	100	2	50	2	50	1	200	4	-	-	-	-	-	-	-	-	50	1	50	1
A5	-	-	200	2	100	2	100	2	500	6	200	2	100	4	200	4	200	4	100	1	50	2
A6	-	-	100	2	50	2	100	2	500	6	100	2	100	3	200	4	100	3	100	1	50	2
A7	-	-	100	2	50	2	100	2	200	4	100	2	100	3	200	4	100	3	50	1	50	2
A8	-	-	100	1	50	1	50	0.5	100	2	-	-	-	-	-	-	-	-	-	-	-	-
A9	-	-	100	1	50	1	50	0.5	100	2	-	-	-	-	-	-	-	-	-	-	-	-
A10	-	-	-	-	-	-	-	-	-	-	100	1	50	1	100	2	100	1	50	1	50	1
Total		**Cost**	1000		550		650		2000		600		450		800		600		550		450	
		Delay	15		15		12		31		8		12		15		12		8		12	

Table 16: Cost and delay values for all activities at decision point D1

Scenario	Probability	Compatibility	Global compatibility
S_1	Pc(1000)=0.25	$C_\alpha(1000;600)=0$	Comp (C_{s1} ;600)=0.4375
	Pc(550)=0.25	$C_\alpha(550;600)=1$	
	Pc(650)=0.25	$C_\alpha(650;600)=0.75$	
	Pc(2000)=0.25	$C_\alpha(2000;600)=0$	
	Pd(15)=0.5	$C_\beta(15;8)=0.125$	Comp (D_{s1};8)=0.1875
	Pd(12)=0.25	$C_\beta(12;8)=0.5$	
	Pd(31)=0.25	$C_\beta(31;8)=0$	
S_2	Pc(600)=0.5	$C_\alpha(600;600)=1$	Comp (C_{s2};600)=0.75
	Pc(450)=0.25	$C_\alpha(450;600)=1$	
	Pc(800)=0.25	$C_\alpha(800;600)=0$	
	Pd(8)=0.25	$C_\beta(8;8)=1$	Comp (D_{s2};8)=0.53125
	Pd(12)=0.5	$C_\beta(12;8)=0.5$	
	Pd(15)=0.25	$C_\beta(15;8)=0.125$	
S_3	Pc(550)=0.5	$C_\alpha(550;600)=1$	Comp (C_{s3};600)=1
	Pc(450)=0.5	$C_\alpha(450;600)=1$	
	Pd(8)=0.5	$C_\beta(8;8)=1$	Comp (D_{s3};8)=0.75
	Pd(12)=0.5	$C_\beta(12;8)=0.5$	

Table 17: Cost and delay probability and compatibility values – D1 dashboard, scenario view

Action	Probability	Compatibility	Global compatibility
a1	Pc(1000)=0.167	$C_\alpha(1000;600)=0$	Comp (C_{a1};600)=0.625
	Pc(550)=0.333	$C_\alpha(550;600)=1$	
	Pc(650)=0.167	$C_\alpha(650;600)=0.75$	
	Pc(2000)=0.167	$C_\alpha(2000;600)=0$	
	Pc(450)=0.167	$C_\alpha(450;600)=1$	
	Pd(15)=0.333	$C_\beta(15;8)=0.125$	Comp (D_{a1};8)=0.375
	Pd(12)=0.333	$C_\beta(12;8)=0.5$	
	Pd(31)=0.167	$C_\beta(31;8)=0$	
	Pd(8)=0.167	$C_\beta(8;8)=1$	
a2	Pc(600)=0.5	$C_\alpha(600;600)=1$	Comp (C_{a2};600)=0.75
	Pc(450)=0.25	$C_\alpha(450;600)=1$	
	Pc(800)=0.25	$C_\alpha(800;600)=0$	
	Pd(8)=0.25	$C_\beta(8;8)=1$	Comp (D_{a2};8)=0.53125
	Pd(12)=0.5	$C_\beta(12;8)=0.5$	
	Pd(15)=0.25	$C_\beta(15;8)=0.125$	

Table 18: Cost and delay probability and compatibility values – D1 dashboard, action view

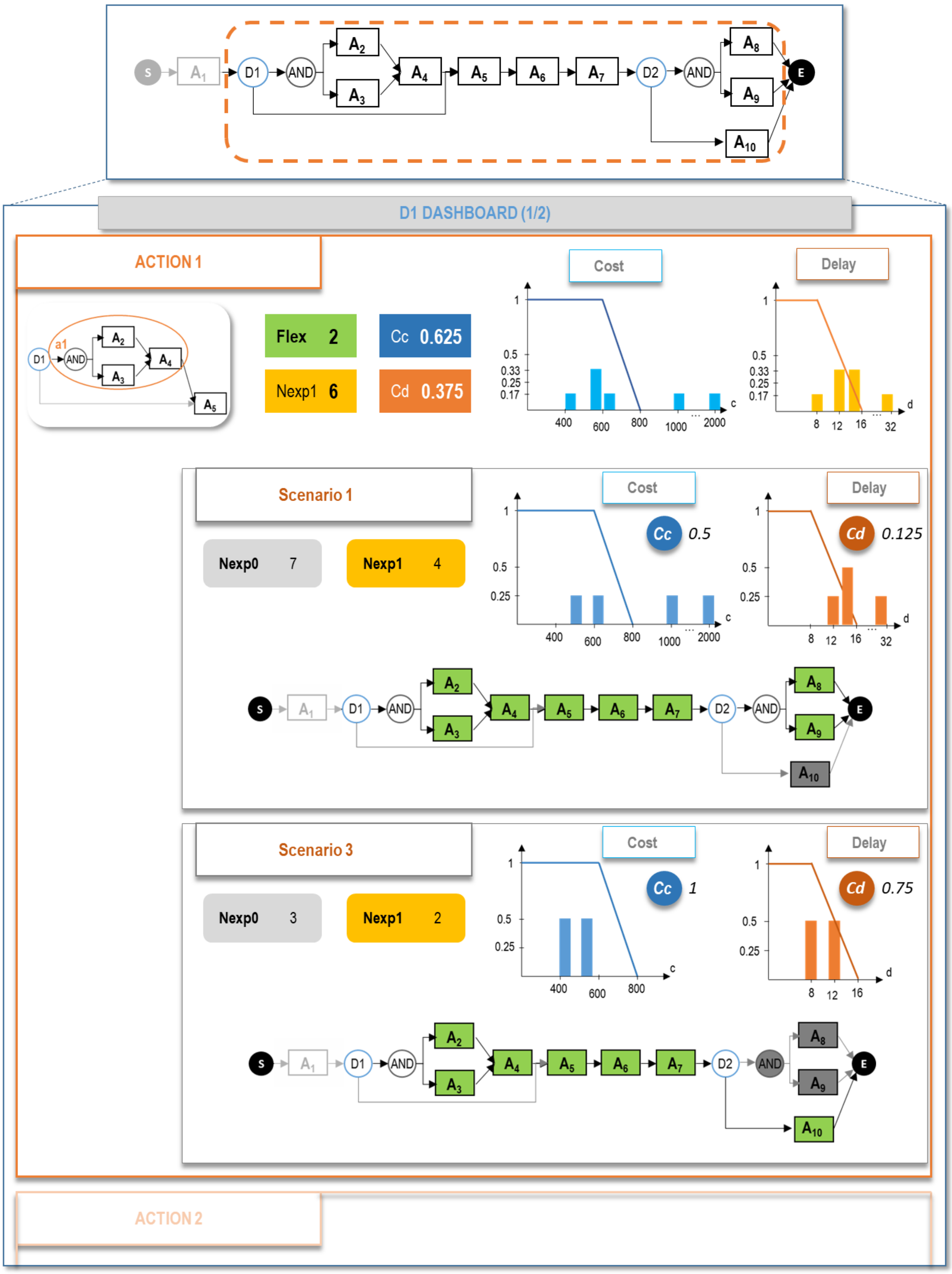


Figure 53: D1 dashboard (1/2)

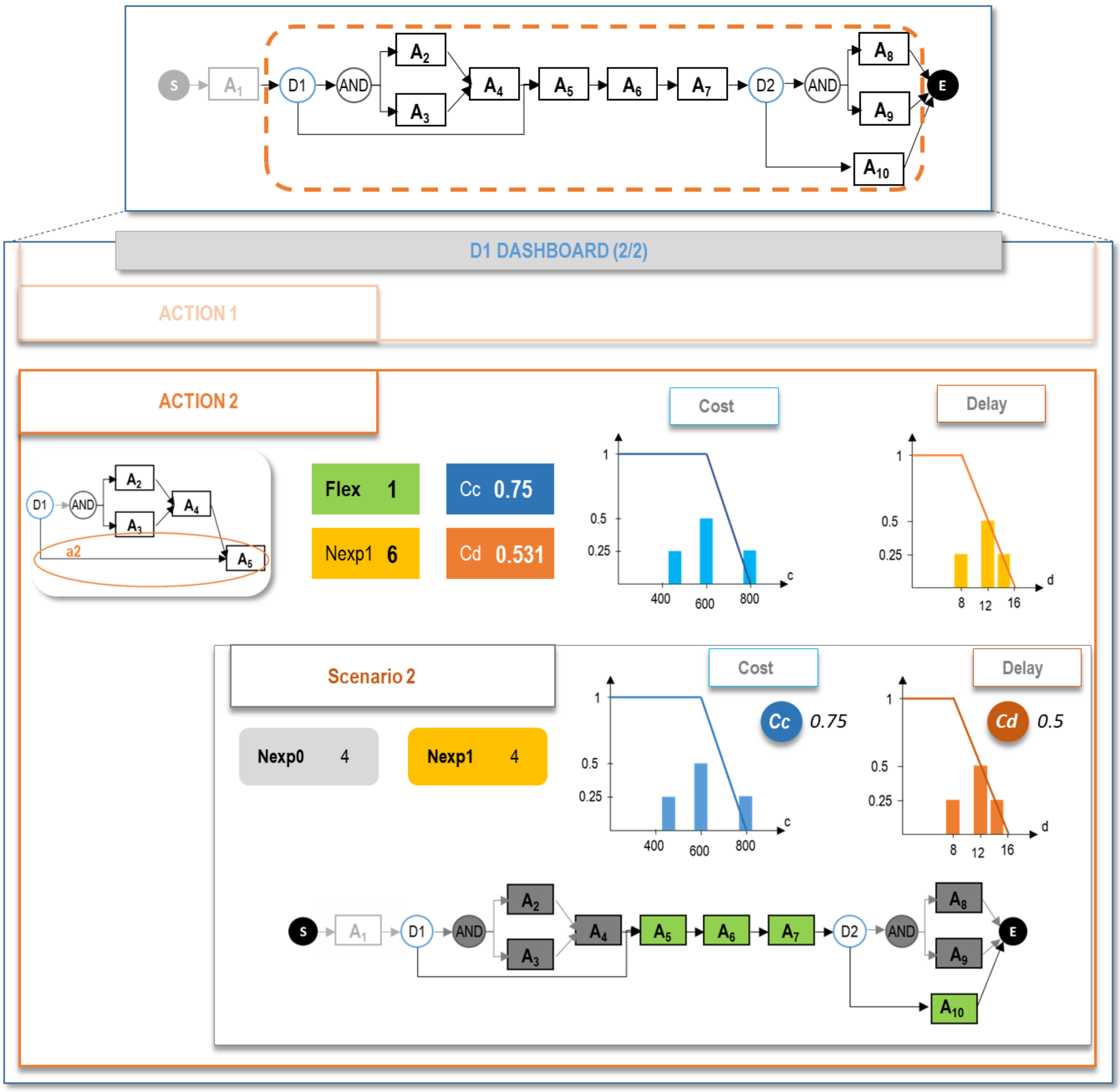


Figure 53 (continuation): D1 dashboard (2/2)

Considering D1 dashboard, the user decides to respect the preferred scenario defined by the process manager. The rationale of the decision should be described, for simplicity reasons it is not illustrated. Action 2 is followed, then the activity A5 (identification of the root causes) is performed. A second version of the process is created indicating this change (Figure 54).

Let us consider that the process is executed following S_2 until the decision-making point D2. Two actions are available in D2, the first one regards scenario S_1, and the second one groups scenarios S_2 and S_3. The same steps are followed in order to compute cost and delay values for the construction of D2 dashboard, which will be proposed to the user to help decision making. Table 19, Table 20 and Table 21 present the updated cost and delay values, and the compute of probability and compatibility for scenario and action views. Considering such values, D2 dashboard is built (Figure 55).

Activity	Real		E_{11}		E_{15}		E_{16}		E_{17}		E_{21}		E_{22}		E_{23}		E_{24}		E_{32}		E_{33}	
	Cost (€)	Delay (hour)	Cost (€)	Delay (hour)	Cost (€)	Delay (hour)	Cost (€)	Delay (hour)	Cost (€)	Delay (hour)	Cost (€)	Delay (hour)	Cost (€)	Delay (hour)	Cost (€)	Delay (hour)	Cost (€)	Delay (hour)	Cost (€)	Delay (hour)	Cost (€)	Delay (hour)
A1	100	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
A2																						
A3																						
A4																						
A5	200	4	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
A6	100	2	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
A7	100	2	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
A8	-	-	100	1	50	1	50	0.5	100	2	-	-	-	-	-	-	-	-	-	-	-	-
A9	-	-	100	1	50	1	50	0.5	100	2	-	-	-	-	-	-	-	-	-	-	-	-
A10	-	-	-	-	-	-	-	-	-	-	100	1	50	1	100	2	100	1	50	1	50	1
Total	Cost		700		600		600		700		600		550		600		600		550		550	
	Delay		11		11		10		13		10		10		11		10		10		10	

Table 19: Cost and delay values for all activities at decision point D2 - normal execution of the process

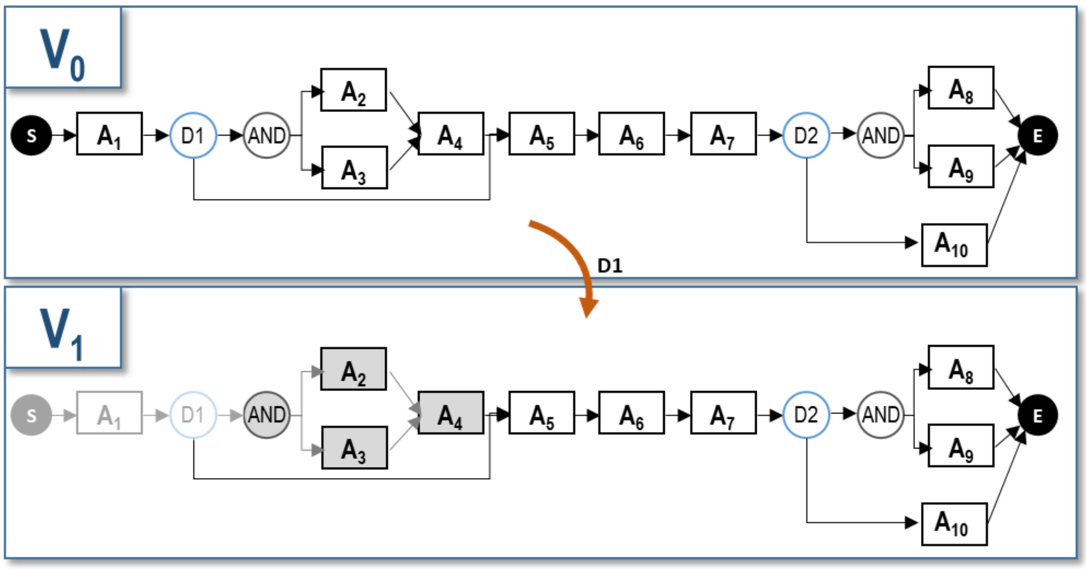


Figure 54: Versions V_0 and V_1 of the process

Scenario	Probability	Compatibility	Global compatibility
S_1	Pc(600)=0.5	C_α(600;600)=1	Comp (C_{s1};600)=0.75
	Pc(700)=0.5	C_α(700;600)=0.5	
	Pd(10)=0.25	C_β(10;8)=0.75	Comp (D_{s1};8)=0.594
	Pd(11)=0.5	C_β(11;8)=0.625	
	Pd(13)=0.25	C_β(13;8)=0.375	
S_{2+3}	Pc(600)=0.5	C_α(600;600)=1	Comp (C_{s2};600)=1
	Pc(550)=0.5	C_α(550;600)=1	
	Pd(10)=0.833	C_β(10;8)=0.75	Comp (D_{s2};8)=0.7291
	Pd(11)=0.167	C_β(11;8)=0.625	

Table 20: Cost and delay probability and compatibility values – D2 dashboard, scenario view- normal execution of the process

Action	Probability	Compatibility	Global compatibility
a1	Pc(600)=0.5	C_α(600;600)=1	Comp (C_{a1};600)=0.75
	Pc(700)=0.5	C_α(700;600)=0.5	
	Pd(10)=0.25	C_β(10;8)=0.75	Comp (D_{a1};8)=0.594
	Pd(11)=0.5	C_β(11;8)=0.625	
	Pd(13)=0.25	C_β(13;8)=0.375	
a2	Pc(600)=0.5	C_α(600;600)=1	Comp (C_{a2};600)=1
	Pc(550)=0.5	C_α(550;600)=1	
	Pd(10)=0.833	C_β(10;8)=0.75	Comp (D_{a2};8)=0.7291
	Pd(11)=0.167	C_β(11;8)=0.625	

Table 21: Cost and delay probability and compatibility values – D2 dashboard, action view- normal execution of the process

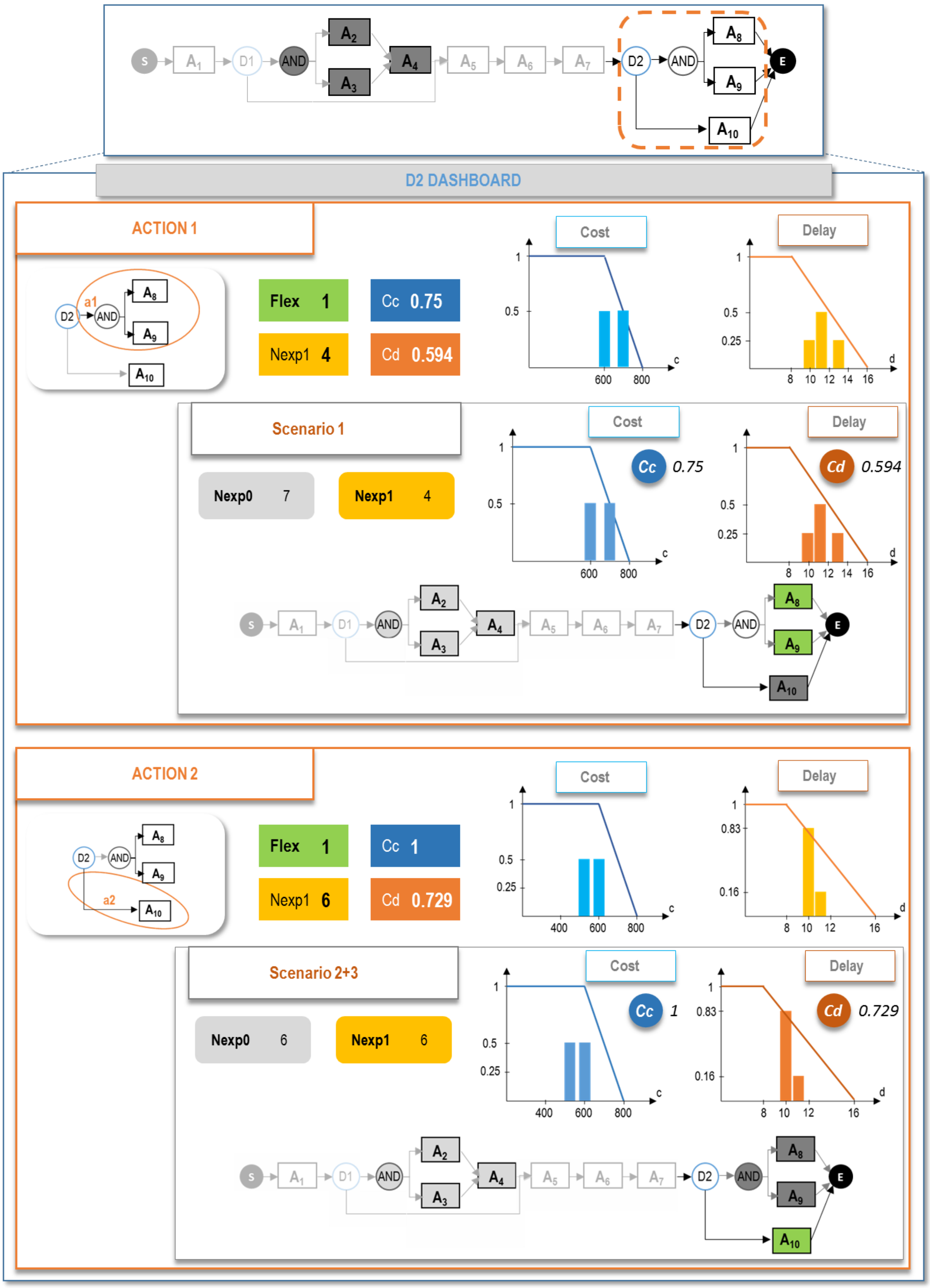


Figure 55: D2 dashboard – normal execution of the process

The user decides to follow action 2 in order to complete the "NC" method. However, one of the values for this action does not respect the constraints set by the stakeholders. The constraint that specifies that delay should not be longer than 8 hours is not satisfied by any of the available values of the delay distribution. Then, this constraint needs to be negotiated before continuing the process. Let us consider that the quality manager decides that this constraint can be released. Then, activity A10 (definition of preventive actions) is performed and the last version of the process is created, as indicates Figure 56.

The process is over and its last version needs to be stored. Step 5 describes the storage step (see section 5.6.1). The next section illustrates another process execution case, when an unexpected event occurs.

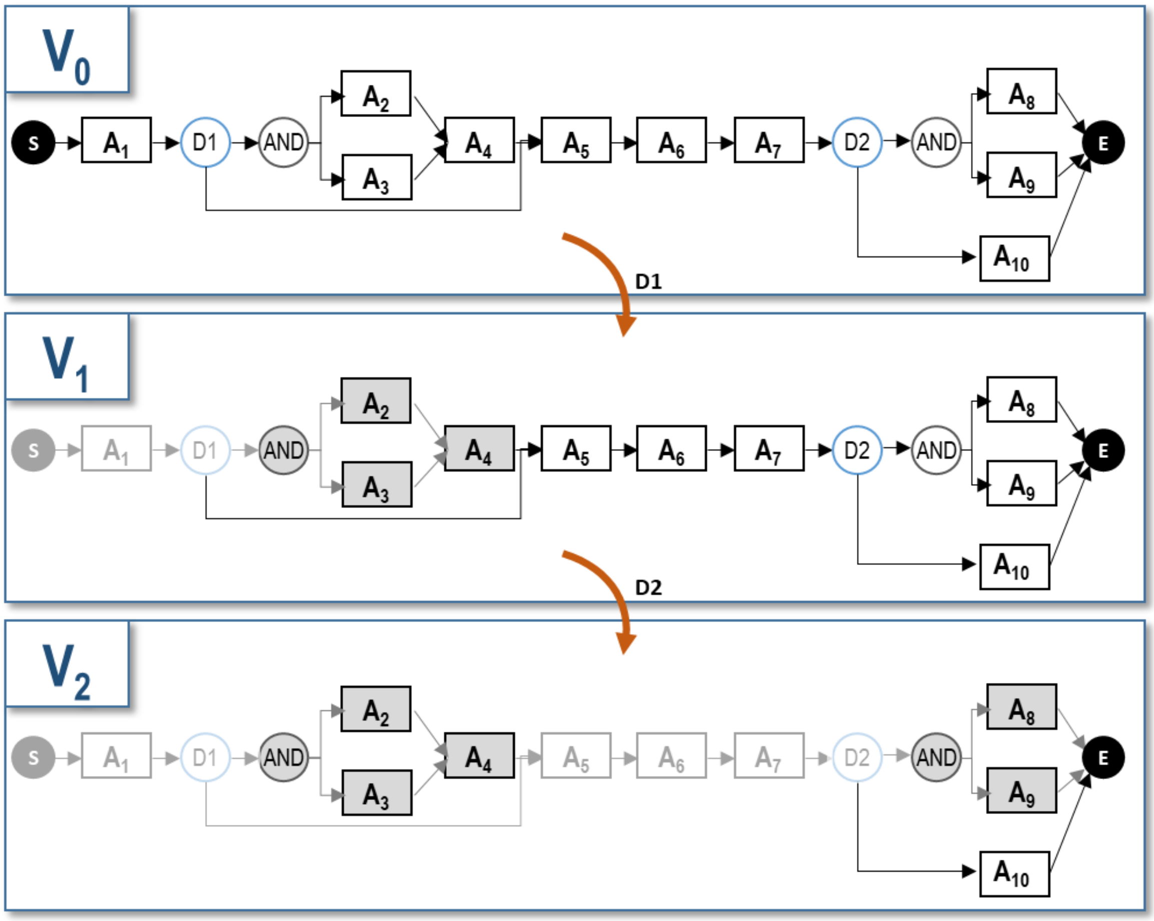


Figure 56: Versions V_0, V_1 and V_2 of the process -– normal execution of the process

5.5.2 Case 2- Execution of the process guided by an unexpected event

Let us consider that the execution of the problem-solving process started in the same manner that the process presented in section 5.5.1, the activity A1 is performed in order to contain the problem. The same actions are achieved: the fallen part is controlled to identify possible damages, the zone where the incident occurred is controlled, and the part is sent to the customer to avoid delays.

Then, decision-making point D1 is reached, and the same decision dashboard illustrated in Figure 53 is proposed to the user. Like in the previous section, the user decides to choose S_2 to continue the process, creating the second version of the process illustrated in Figure 54. Then, activity A5, the determination of the root causes of the problem needs to be performed. While information was being collected in order to define the root cause of the problem (A5), the customer contacts the quality department of the company to report that the received part does not respect the painting requirements and that it will be sent back to the company. This unexpected event affects the execution of the agile process since it needs an immediate treatment. A decision-making point based on event (De1) is immediately created in order to define what to do next (activity A5 is set as “unfinished”).

It is decided that the most urgent action to take is to repair the part in order to deliver it to the customer in order to avoid further delays. For this purpose, the activity A11: “Definition of corrective actions” is created. It considers: First, the application of non-destructive testing on the part to verify that the problem is only painting-related; Second, controlling the painting to determine possible gaps with the requirement; and Third, preparing and repainting the part. Decision makers decide that, once A11 is over, A5 can be resumed and the process can go back to its normal structure (A6, A7, etc.). Then, a new version of the process is created, as illustrated in Figure 57.

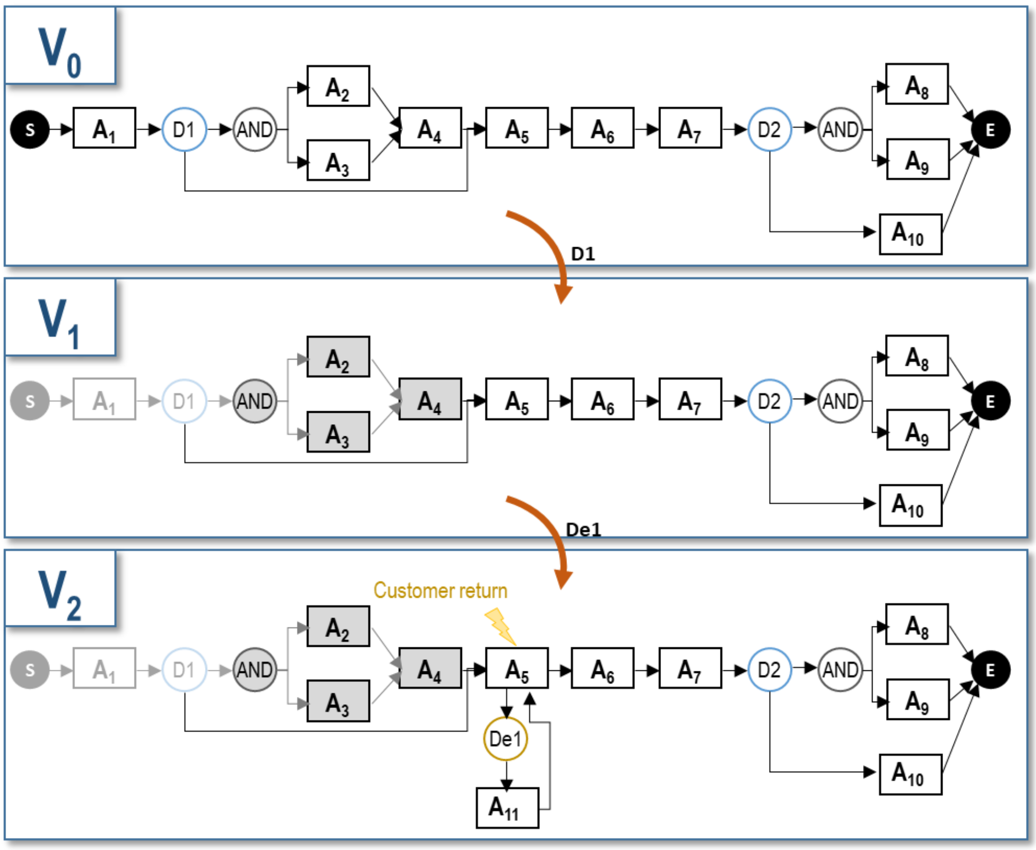


Figure 57: Versions V_0, V_1 and V_2 of the process – unexpected event

Activity	Real		E_{11}		E_{15}		E_{16}		E_{17}		E_{21}		E_{22}		E_{23}		E_{24}		E_{32}		E_{33}	
	Cost (€)	Delay (hour)	Cost (€)	Delay (hour)	Cost (€)	Delay (hour)	Cost (€)	Delay (hour)	Cost (€)	Delay (hour)	Cost (€)	Delay (hour)	Cost (€)	Delay (hour)	Cost (€)	Delay (hour)	Cost (€)	Delay (hour)	Cost (€)	Delay (hour)	Cost (€)	Delay (hour)
A1	100	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
A2																						
A3																						
A4																						
A5	230	5	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
A11	150	3	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
A6	100	2	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
A7	100	2	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
A8	-	-	100	1	50	1	50	0.5	100	2	-	-	-	-	-	-	-	-	-	-	-	-
A9	-	-	100	1	50	1	50	0.5	100	2	-	-	-	-	-	-	-	-	-	-	-	-
A10	-	-	-	-	-	-	-	-	-	-	100	1	50	1	100	2	100	1	50	1	50	1
Total	**Cost**		880		780		780		880		780		730		780		780		730		730	
	Delay		15		15		14		17		14		14		15		14		14		14	

Table 22: Cost and delay values for all activities at decision point D2- unexpected event

Let us now consider that activities A11, A5, A6 and A7 have been performed and that decision-making point D2 is reached. Table 22, Table 23 and Table 24 present the updated values of cost, delay, probability and compatibility for the process. In Table 22, the row corresponding to A11 is incorporated. The values presented in the last row of Table 22 show that the unexpected event impacted the process by increasing cost and delay global values.

Scenario	Probability	Compatibility	Global compatibility
S_1	Pc(880)=0.5	$C_\alpha(880;600)=0$	Comp $(C_{s1};600)=0.05$
	Pc(780)=0.5	$C_\alpha(780;600)=0.1$	
	Pd(15)=0.5	$C_\beta(15;8)=0.125$	Comp $(D_{s1};8)=0.563$
	Pd(14)=0.25	$C_\beta(14;8)=0.25$	
	Pd(17)=0.25	$C_\beta(17;8)=0$	
S_2	Pc(780)=0.75	$C_\alpha(780;600)=0.1$	Comp$(C_{s2};600)=0.1625$
	Pc(730)=0.25	$C_\alpha(730;600)=0.35$	
	Pd(14)=0.75	$C_\beta(14;8)=0.25$	Comp $(D_{s2};8)=0.21875$
	Pd(15)=0.25	$C_\beta(15;8)=0.125$	
S_3	Pc(730)=1	$C_\alpha(730;600)=0.35$	Comp $(C_{s3};600)=0.35$
	Pd(14)=1	$C_\beta(14;8)=0.25$	Comp $(D_{s3};8)=0.25$

Table 23: Cost and delay probability and compatibility values – D2 dashboard, scenario view-unexpected event

Action	Probability	Compatibility	Global compatibility
a1	Pc(880)=0.5	$C_\alpha(880;600)=0$	Comp $(C_{a1};600)=0.05$
	Pc(780)=0.5	$C_\alpha(780;600)=0.1$	
	Pd(15)=0.5	$C_\beta(15;8)=0.125$	Comp $(D_{a1};8)=0.563$
	Pd(14)=0.25	$C_\beta(14;8)=0.25$	
	Pd(17)=0.25	$C_\beta(17;8)=0$	
a2	Pc(780)=0.5	$C_\alpha(780;600)=0.1$	Comp $(C_{a2};600)=0.225$
	Pc(730)=0.5	$C_\alpha(730;600)=0.35$	
	Pd(14)=0.833	$C_\beta(14;8)=0.25$	Comp $(D_{a2};8)=0.229$
	Pd(15)=0.167	$C_\beta(15;8)=0.125$	

Table 24: Cost and delay probability and compatibility values – D2 dashboard, action view-unexpected event

Considering such information, D2 dashboard can be constructed (Figure 58). The cost and delay values for both actions (a1 and a2) do not respect the constraints set by the stakeholders. The constraint that specifies that delay should not be longer than 8 hours is not satisfied by any of the available values of the delay distributions. Moreover, the constraint specifying that cost should not be more than 600€ is not satisfied by any of the available values of the cost distributions. Then, these constraints need to be negotiated before continuing the process. Let us consider that the quality manager decides that both constraints can be released. Then, activity A10 (definition of preventive actions) is performed and the last version of the process is created, as indicates Figure 59.

Once that the process has been completely carried out, it is stored in the knowledge and/or experience base. The storage step is detailed in the next section.

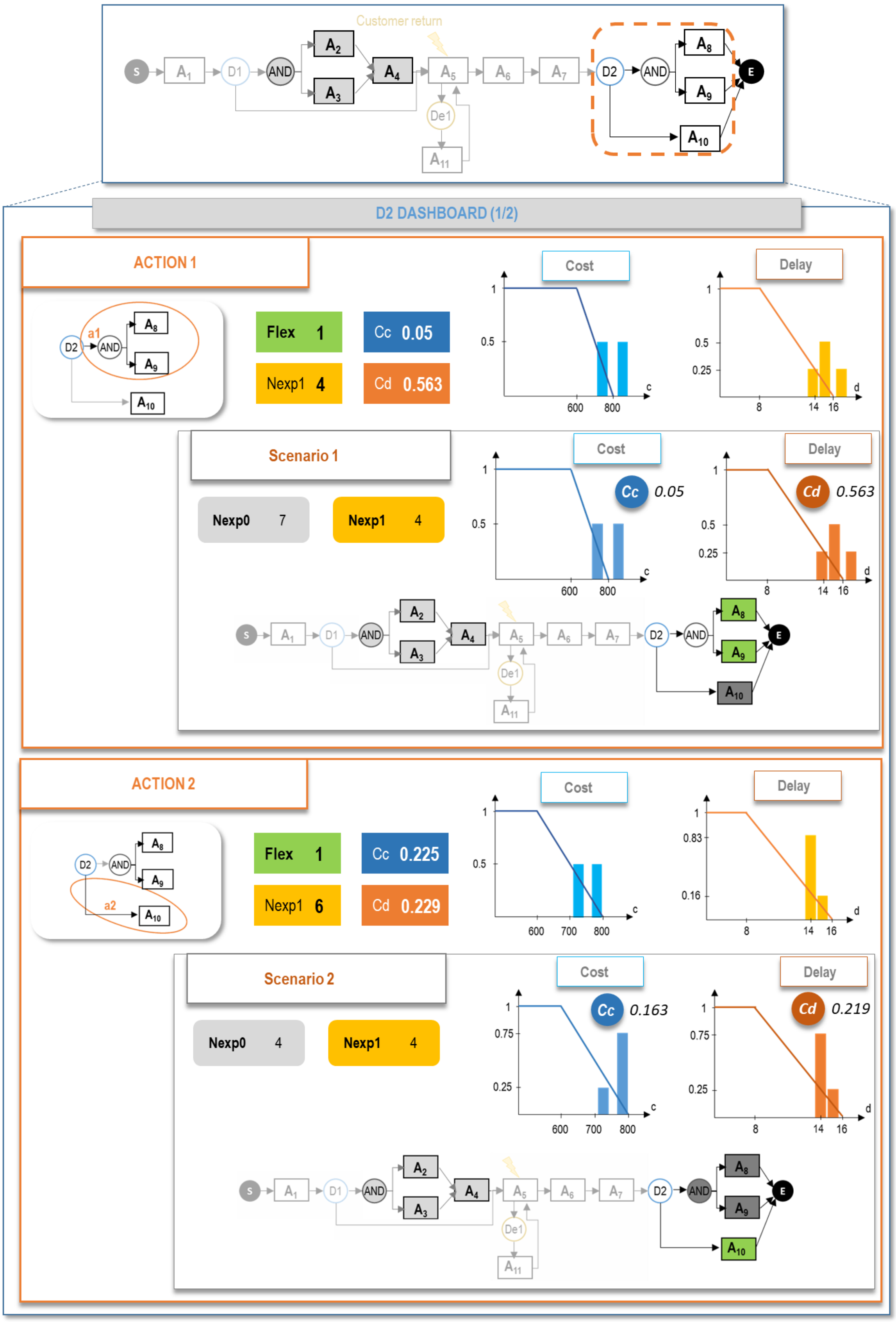


Figure 58: D2 dashboard - – unexpected event (1/2)

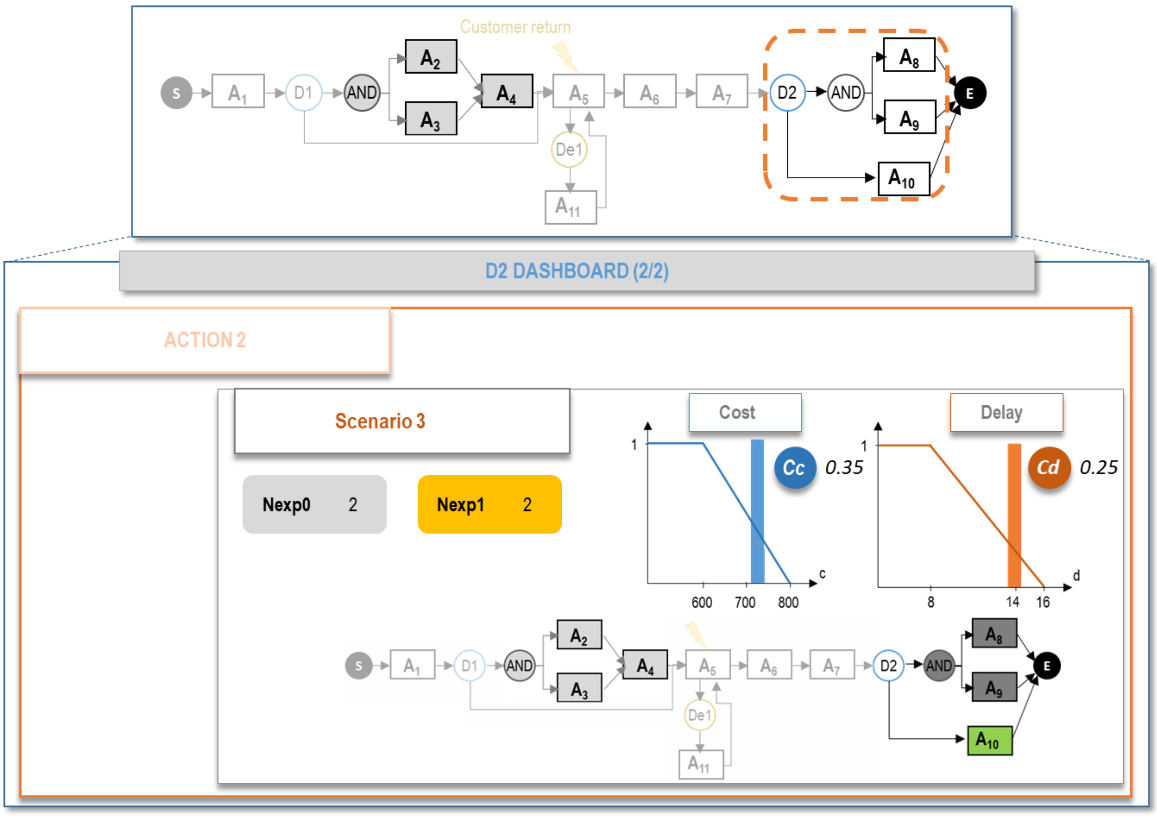


Figure 58 (continuation): D2 dashboard – unexpected event (2/2)

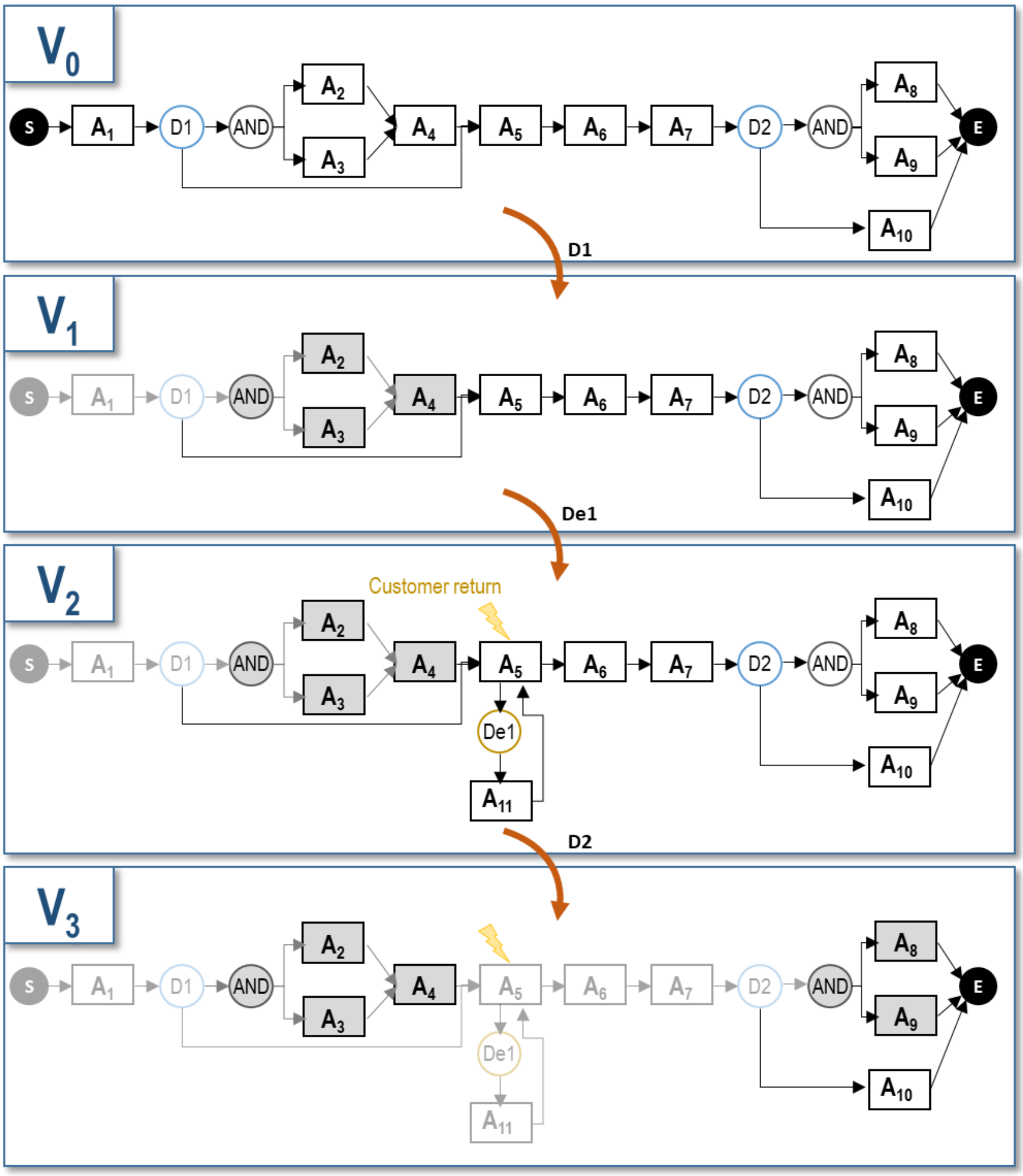


Figure 59: Versions V_0, V_1, V_2 and V_3 of the process

5.6 Step 5. Storage in the Knowledge/experience base

The last version of the process needs to be stored in the experience base and, if the structure of the process has changed, in the knowledge base. The storage step is described in this section. Both cases described in section 5.5, normal execution of the process and occurrence of an unexpected event, are treated separately in sections 5.6.1 and 5.6.2.

5.6.1 Case 1- Normal execution of the process

The process manager considers the last version of the process to define if there is a change in the structure of the process. As illustrated in Figure 60, the structure of the last version of the process corresponds to the structure of scenario S_2. It means that the current process is stored into the experience base as a new experience linked to S_2. Then, as shown in Table 25, the experience base is updated with the new experience, which is now available for its future reuse.

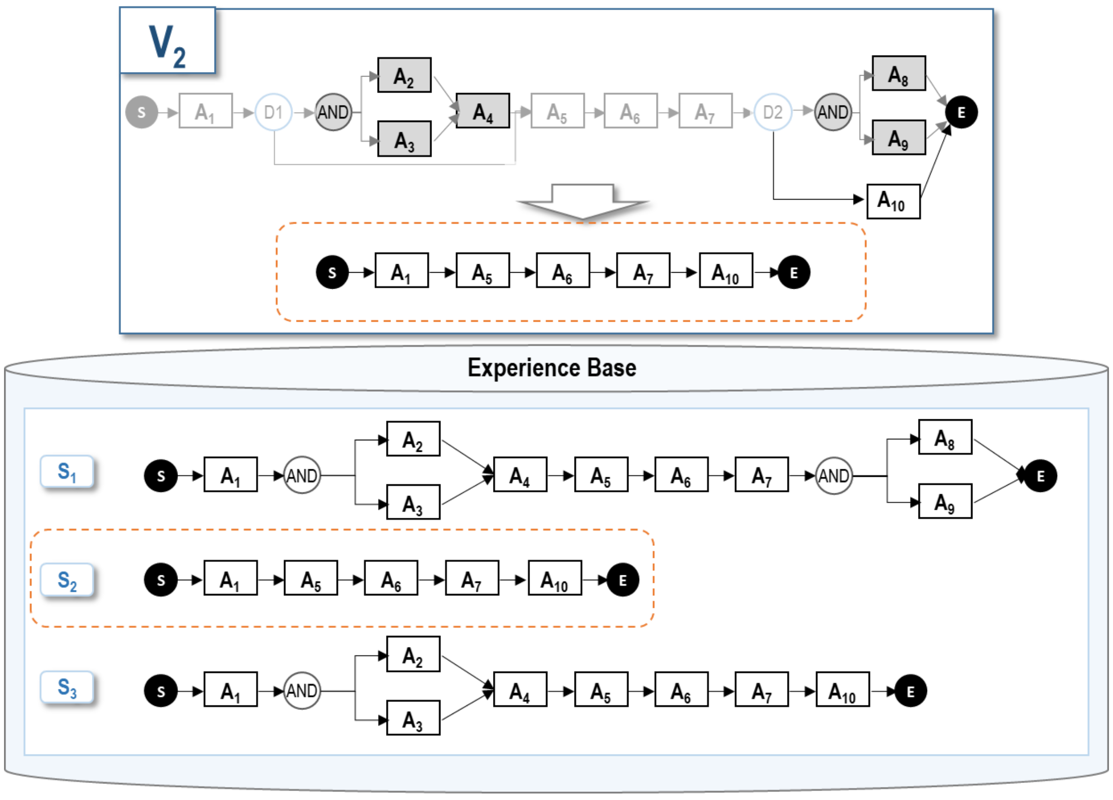


Figure 60: Last version of the process and Experience base

Scenario	Experience	T1	T2	T3	T4
S_1	E_{11}	8D	Quality	Big panel	Part falling
	E_{12}	8D	Quality	Small panel	Tooling
	E_{13}	8D	Quality	Big panel	Dimensional
	E_{14}	8D	Quality	Big panel	Painting
	E_{15}	8D	Quality	Small part	Electrical crash
	E_{16}	8D	Quality	Big part	Dimensional
	E_{17}	8D	Quality	Chemical milling line	Line stoppage

S_2	E_{21}	NC	Quality	Big part	Dimensional
	E_{22}	NC	Quality	Small part	Painting
	E_{23}	NC	Quality	Small part	Dimensional
	E_{24}	NC	Quality	Small panel	Part falling
	E_{25}	NC	Quality	Big part	Part falling
S_3	E_{31}	8D	Quality	Small part	Dimensional
	E_{32}	8D	Quality	Big part	Painting
	E_{33}	8D	Quality	Small panel	Part falling

Table 25: Updated experiences tags

5.6.2 Case 2- Execution of the process guided by an unexpected event

When an unexpected event has occurred, in order to store the process, the attributes of the nodes need to be updated. First, the activities impacted by the occurrence of the event, denominated "*Event$_1$*", are identified. The impacted activities are all those specially created because of the event, and those which were performed after the event. For this process such activities are: A11, A5, A6, A7, and A10. For those activities, their attributes need to be updated including the event "*Event$_1$*" into the event list attributes. The list of events for activities A5, A6, A7 and A10 contained the event "*None*" (the activities were activated during the standard execution of the process), and, now, event "*Event$_1$*". However, A11 only contains the event "*Event$_1$*" in the event list attributes, which means that the activity is inhibited during the standard execution of the process. Then, if "Event$_1$" reoccurs, all impacted activities will become "activated" in order to help decision making. Moreover, the decision-making point based on event De1 is stored into the knowledge base as a XOR node. Such node only contains the event "*Event$_1$*" in the event list attributes.

In order to store the process, its structure is considered. It does not correspond to any existing scenario in the knowledge base. Then, a new scenario S_4 needs to be created. The knowledge base is updated with the new scenario, as illustrated in Figure 61. The process is also stored as a new experience, linked to S_4, into the experience base (lower part of Figure 61).

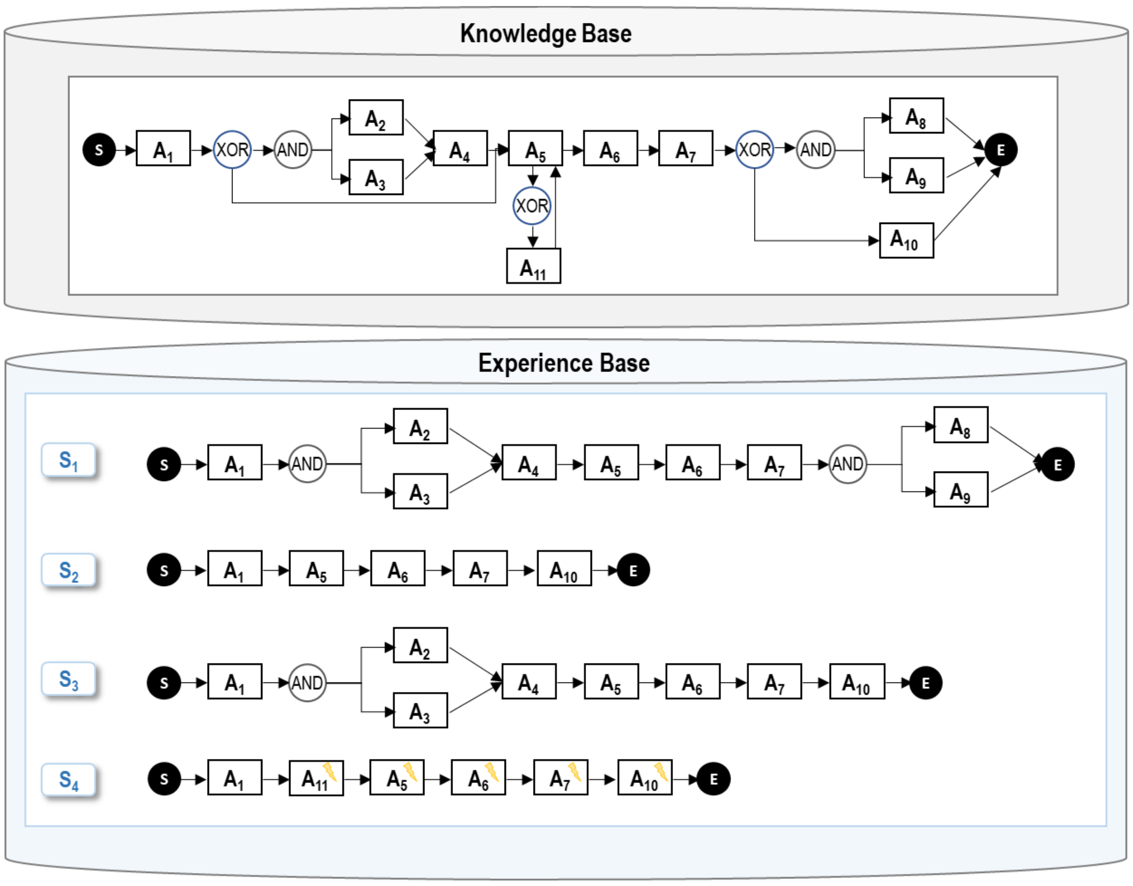


Figure 61: Updated knowledge and experience base

5.7 Conclusion

A real case conducted at a surface treatment company was studied in this chapter and used to inspire an illustration of the proposed approach. It enhanced comprehension of all the concepts from the agile model. Information collected from interviews and from existing data bases were analysed in order to create the knowledge and experience base of the company. For this purpose, experiences were given with tags and indicators. Once that the knowledge and experience base were built, a new case was treated following the agile lifecycle.

First, the problem was characterized through a set of tags. Also, the process indicators were defined. Second, the experience filtering step was conducted taking into consideration the defined target tags. The first version of the process was then defined and adapted by the process manager. During process execution two cases were described. First, the standard execution of the process was considered. Second, the treatment of an unexpected event was detailed. Finally, the storage into the knowledge and experience bases was defined for both situations.

The complete knowledge and experience base could be used as a basis for the company. Moreover, the complete agile lifecycle could be incorporated into the company problem-solving strategy. This would improve problem-solving processes through the addition of agility to better adapt each process to its context and to respond to change when needed.

6. CONCLUSION

This thesis work focuses on the lack of agility of problem-solving processes in nowadays organisations. In order to tackle such problem, an agile problem-solving process model based on experience and knowledge capitalization and reuse is proposed and applied to a company real case. The concepts introduced in this document are synthesized below.

In order to clearly state the objectives of the research, a literature review was conducted. The three main research pillars were introduced in chapter 2. First, the concepts of agility in different application domains were presented, and their key characteristics and principles were considered and adapted to define an agile process. Second, knowledge and experience capitalization and reuse principles were introduced. An agile process is guided by such mechanisms, which allows the capitalization of experiences and their reuse in order to support future decision making. Third, problem solving concepts and methods were outlined. Such processes are taken into consideration in this work because of their dual nature (mixed structured and exploratory activities). Three of the most used methods in industry were used: PDCA, 8D, DMAIC. It has been concluded that agile principles could help problem-solving processes to be more flexible and adapt to changes in their environment. Also, while knowledge and experiences capitalization and reuse principles are currently applied to problem-solving processes (during the standardization step), they should be simplified in order to reinforce knowledge and experiences reuse.

The theoretical bases of the model were introduced in chapter 3. An agile process was defined. Its key elements are activities, decision-making points, versioning system, and roles. The knowledge and experience capitalization and reuse mechanisms were presented followed by the concepts of types of processes, scenario and experience. The agile process characterization elements, the tagging system and indicators were described.

In chapter 4, the agile lifecycle, based on CBR principles, was proposed. The five steps that allow the definition of the agile process, its execution until its storage into the knowledge and experience bases were described. The first step regards the characterization of the process using a dedicated tagging system and the definition of constraints. The second step considers the retrieval and filtering of past experience using semantical similarity, then, only experiences considered similar to the current process are retrieved for further steps. In step three, a first version of the process is built using the complete process model from the knowledge base, which is adapted using a decision support dashboard. The fourth step regards the execution of the agile process. Using the first version of the process as a guideline, at each decision-making point, decision makers decide the way forward supported by decision dashboards. In the fifth and last step, the process is stored into the experience and knowledge bases. If the process structure matches an existing scenario, it is stored as a new experience. If it does not match, the knowledge expert needs to validate the new scenario to be added into the knowledge base. During each step, an illustrative example was described in order to clarify the concepts.

In chapter 5, a real case application was described. The case study was conducted into the quality department of a surface treatment company. Problem solving data was collected through interviews and data-base consultation in order to build the company knowledge and experience bases. Considering such information, a new problem was defined. The problem-solving process was characterized through a set of tags and indicators (step 1). Experience filtering was performed considering the current process tags and the past experience tags (step 2). The first version of the process was adapted by the process manager supported by the decision dashboard (step 3). Such a first version was followed by the user in order to execute the process (step 4). During the step four, two cases were described. The first one regards the execution of a standard problem-solving process, and the second one describes the treatment of an unexpected event that occurred during the process execution. Finally, the process is stored into the knowledge and experience base (step 5), for the two cases described in step 4.

The proposed agile problem solving model contributes to the problem described in the first chapter of this work regarding the lack of agility of most problem-solving processes. Our model provides agility to problem-solving processes through supported decision making and reconfiguration capabilities. The defined agile lifecycle allows users to retrieve past similar situations in order to define and/or adapt their current process according to the best practices already performed within the company. The process execution is continuously guided and supported, mostly during decision-making points, due to relevant information compiled into decision-dashboards. Moreover, the agile lifecycle ensures that when the process has been carried out it is properly stored in the KB/EB in order to ensure its future reuse.

The application of the agile problem solving model contributed to its improvement and to the definition of some limits of our work. The perspectives and further work for our research were defined taking into consideration such limits. Perspectives are related to the methodology proposed by the model, the mechanisms and algorithms necessary to deploy such methodology and the application of the model to industrial companies. They are listed next according to an estimated horizon: short, medium and long-term.

Four perspectives were identified to be deployed in the short term:

- One of the agile principles most applied in nowadays companies is agile management. Some of the most widespread practices consider that teams are self-organized, work in autonomy and a comfortable work environment will lead to better results. A first approach to agile management was considered in this work. For instance, it has been said that when users execute the process, they use the first version defined by the process manager as a guideline. Then, the process manager states the guidelines for the work and the team work in autonomy. However, more work could be done to include further agile management principles to problem-solving processes.

- The advice of experts should be better integrated into the knowledge/experience bases. For instance, regarding scenarios cost and delay values, there is a special case that needs to be clarified and properly defined. It has been said that the cost and delay values of a scenario considered to compute V_0, correspond to probability distributions of past experiences. However, when the experience base is empty, i.e. no process has ever been performed before, the process manager needs to define *expected* cost and delay values for each scenario based on an expert criteria. Then, V_0 dashboard would be computed from those nominal cost and delay values.
- The state of an activity was considered as finished or unfinished in this work. Moreover, when an unexpected event occurs during the execution of an activity, such activity can be resumed after treating the unexpected event, or the next activity can be performed. In the case where the same activity is resumed, it has been considered that the activity was unfinished. However, the amount of completion of the activity was not considered. Further work needs to be done to determine an algorithm that integrates the activity progress. Then, the activity could be characterized in a more accurately manner, and it could be resumed at the exact same point that it was stopped by the unexpected event.
- A major point of improvement of the agile problem solving model could be to collect feedback regarding its implementation in industry. It could be interesting to analyse comments and questions arising from users when they implement the complete agile lifecycle.

The two medium-term perspectives linked to this work are described next:

- A multicriteria probability could be computed for each scenario in order to class them and to determine the most appropriate one with regards to the current process. For this purpose, several criteria could be considered such as the cost and delay of the process (already used in this study). Then, at each decision point, all criteria could be weighted in order to select in the first place the most suitable scenario, and then a classification of the remaining scenarios.
- A direct application of the agile problem solving model could be to apply its principles into the existing problem solving platform ProWhy. A new module of the software could be created including the agile lifecycle steps. Then, the process could be properly characterized into the tool, similar previous experiences could be proposed by the tool to create a V_0, the process execution could be followed through the software interface and finally, the problem-solving process could be stored into ProWhy database.

Finally, three long-term perspectives were identified for further works.

- One of the ten agility key concept is collaboration. As it has been said in chapter 2, even if collaboration was considered out of the scope of this research work, it is an interesting perspective for further work. Agile processes should be collaborative, not only within the company perimeter but also with customers and suppliers. Sharing information between actors could be useful to anticipate further problems and risks, and the possibility to work in a cooperative way could enable faster and more efficient resolution of problems. Moreover, a PhD thesis is being performed between Axsens-bte and LGP-ENIT regarding the subject of collaboration.
- Risk analysis could be performed within the agile process. When users arrive to a decision-making point, the probability of occurrence of unexpected events could be added into the decision dashboard based on past experiences. This could be done by considering information available in the knowledge base such as the scenarios linked to a given event.
- The agile process model introduced in this work is focused on problem-solving processes due to their dual (structured/exploratory) nature. However, authors believe that the model can be generalized to other types of business processes. The model simplicity and user-focused elements could help to define other agile processes that could follow an adapted agile lifecycle. For instance, the model could be applied to two types of process: i) new product development processes which are performed in an innovation context, then, agility could be used to improve the alternate stable/instable periods of development (Wieder et al., 2007), ii) the bidding process (Sylla et al., 2017), where incremental versions of the final document are created in order to define the final version corresponding to the offer for the potential customer.

PUBLICATIONS RELATED TO THIS PHD WORK

Llamas V., Coudert T., Geneste L., Romero Bejarano J. C. and De Valroger A., Experience reuse to improve agility in knowledge-driven industrial processes, IEEE International Conference on Industrial Engineering and Engineering Management (IEEM'16), Bali, Indonesia, December 4-7, 2016, pp. 651-655.

Llamas V., Coudert T., Geneste L., Romero Bejarano J.C., De Valroger A., Proposition of an agile knowledge-based process model, 8th Internatial conference on Manufacturing Modelling, Management, and Control, MIM'2016, Volume 49, Issue 12, 2016, Pages 1092-1097, 28-30 june, 2016, Troyes.

REFERENCES

A

Aalst, W.M.P. van der, Hofstede, A.H.M. ter, Weske, M., 2003. Business Process Management: A Survey, in: Aalst, W.M.P. van der, Weske, M. (Eds.), Business Process Management, Lecture Notes in Computer Science. Springer Berlin Heidelberg, pp. 1–12.

Aamodt, A., Plaza, E., 1994. Case-Based Reasoning - Foundational Issues, Methodological Variations, and System Approaches. Ai Commun. 7, 39–59.

Abrahamsson, P., Warsta, J., Siponen, M.T., Ronkainen, J., 2003. New directions on agile methods: a comparative analysis, in: 25th International Conference on Software Engineering, 2003. Proceedings. Presented at the 25th International Conference on Software Engineering, 2003. Proceedings., pp. 244–254.

Agarwal, A., Shankar, R., Tiwari, M.K., 2007. Modeling agility of supply chain. Ind. Mark. Manag. 36, 443–457.

Agarwal, A., Shankar, R., Tiwari, M.K., 2006. Modeling the metrics of lean, agile and leagile supply chain: An ANP-based approach. Eur. J. Oper. Res. 173, 211–225.

Alvarado, M., Bañares-Alcántara, R., Trujillo, A., 2005. Improving the Organisational Memory by recording decision making, rationale and team configuration. J. Pet. Sci. Eng., Intelligent Computing in Petroleum Engineering 47, 71–88.

B

Balaji, M., Velmurugan, V., Sivabalan, G., Ilayaraja, V.S., Prapa, M., Mythily, V., 2014. ASCTM Approach for Enterprise Agility. Procedia Eng., "12th Global Congress on Manufacturing and Management" GCMM - 2014 97, 2222–2231.

Battistella, C., De Toni, A.F., De Zan, G., Pessot, E., 2017. Cultivating business model agility through focused capabilities: A multiple case study. J. Bus. Res. 73, 65–82.

Beck, K., 2000. Extreme Programming Explained: Embrace Change. Addison-Wesley Professional.

Béler, C., 2008. Modélisation générique d'un retour d'expérience cognitif. Application à la prévention des risques (phd).

Béler, C., Desforges, X., 2007. EXPERIENCE FEEDBACK, FROM CASES TO KNOWLEDGE. IFAC Proc. Vol., 4th IFAC Conference on Management and Control of Production and Logistics 40, 43–48.

Bergmann, R., 2002. Experience management: Foundations, development methodology, and Internet-based applications. Springer.

Bergmann, R., Gil, Y., 2014. Similarity assessment and efficient retrieval of semantic workflows. Inf. Syst. 40, 115–127.

Bhote, K.R., 1991. World Class Quality: Les 7 outils Shainin de la qualité. Dunod, Paris.

Boehm, B.W., 1988. A Spiral Model of Software Development and Enhancement. Computer 21, 61–72.

Boon Sin, A., Zailani, S., Iranmanesh, M., Ramayah, T., 2015. Structural equation modelling on knowledge creation in Six Sigma DMAIC project and its impact on organizational performance. Int. J. Prod. Econ. 168, 105–117.

C

Cambridge University Press, 2017. Cambridge English Dictionary. URL http://dictionary.cambridge.org/dictionary/english/ (accessed 5.4.17).

Choo, A.S., Nag, R., Xia, Y., 2015. The role of executive problem solving in knowledge accumulation and manufacturing improvements. J. Oper. Manag. 36, 63–74.

Christopher, M., 2000. The Agile Supply Chain: Competing in Volatile Markets. Ind. Mark. Manag. 29, 37–44.

Conboy, K., Fitzgerald, B., 2004. Toward a Conceptual Framework of Agile Methods, in: Extreme Programming and Agile Methods - XP/Agile Universe 2004. Presented at the Conference on Extreme Programming and Agile Methods, Springer, Berlin, Heidelberg, pp. 105–116.

Czinki, A., Hentschel, C., 2016. Solving Complex Problems and TRIZ. Procedia CIRP, Structured Innovation with TRIZ in Science and Industry: Creating Value for Customers and Society 39, 27–32.

D

Dalkir, K., 2013. Knowledge Management in Theory and Practice. Routledge.

Davenport, T.H., 1993. Process Innovation: Reengineering Work Through Information Technology. Harvard Business School Press, Boston, MA, USA.

de Mast, J., Lokkerbol, J., 2012. An analysis of the Six Sigma DMAIC method from the perspective of problem solving. Int. J. Prod. Econ., Compassionate Operations 139, 604–614.

Debauche, B., Mégard, P., 2004. BPM Business Process Management : Pilotage métier de l'entreprise. Hermes Science Publications, Paris.

Deming, W.E., 2000. The New Economics for Industry, Government, Education, 2nd edition. ed. MIT Press, Cambridge.

Dove, R., 1994. Tools for Analyzing and Constructing Agility.

Dubois, D., Fargier, H., Prade, H., 1996. Possibility theory in constraint satisfaction problems: Handling priority, preference and uncertainty. Appl. Intell. 6, 287–309.

Duret, D., Pillet, M., 2011. Qualité en production: De l'ISO 9000 à Six Sigma. Editions Eyrolles.

F

Fey, V.R., Rivin, E.I., Vertkin, I.M., 1994. Application of the Theory of Inventive Problem Solving to Design and Manufacturing Systems. CIRP Ann. - Manuf. Technol. 43, 107–110.

Fowler, M., Highsmith, J., 2001. The Agile Manifesto. Softw. Dev. 28–32.

G

Gill, A.Q., 2015. Agile enterprise architecture modelling: Evaluating the applicability and integration of six modelling standards. Inf. Softw. Technol. 67, 196–206.

Goldman, S.L., 1995. Agile Competitors and Virtual Organizations: Strategies for Enriching the Customer. Van Nostrand Reinhold.

Gruber, T.R., 1993. A translation approach to portable ontology specifications. Knowl. Acquis. 5, 199–220.

H

Highsmith, J., 2013. Adaptive Software Development: A Collaborative Approach to Managing Complex Systems. Addison-Wesley.

I

IAQG (International Aerospace Quality Group), 2016. EN 9136 - Aerospace Series – Root Cause Analysis and Problem Solving (9S Methodology). ASD-STAN.

IAQG (International Aerospace Quality Group), 2014. Chapter 7.4: Root Cause Analysis and Problem Solving, in: Supply Chain Management Handbook (SCMH).

Ilevbare, I.M., Probert, D., Phaal, R., 2013. A review of TRIZ, and its benefits and challenges in practice. Technovation 33, 30–37.

International Organizational for Standardization, 2015. ISO 9001: Quality management systems — Requirements.

Izza, S., Imache, R., Vincent, L., Lounis, Y., 2008. An approach for the evaluation of the agility in the context of enterprise interoperability. Springer, New York.

J

Jabrouni, H., 2012. Exploitation des connaissances issues des processus de retour d’expérience industriels (phd).

Jabrouni, H., Kamsu-Foguem, B., Geneste, L., Vaysse, C., 2011. Continuous improvement through knowledge-guided analysis in experience feedback. Eng. Appl. Artif. Intell., Semantic-based Information and Engineering Systems 24, 1419–1431.

James, M., 2010. Scrum Reference Card | Scrum Reference Card.

K

Kamsu Foguem, B., Coudert, T., Béler, C., Geneste, L., 2008. Knowledge formalization in experience feedback processes: An ontology-based approach. Comput. Ind., Enterprise Integration and Interoperability in Manufacturing Systems 59, 694–710.

Kast, R., 1993. La théorie de la décision. La Découverte.

Katayama, H., Bennett, D., 1999. Agility, adaptability and leanness: A comparison of concepts and a study of practice. Int. J. Prod. Econ. 60–61, 43–51.

Kepner, C., Tregoe, B., 1982. The new rational manager. John Martin Publishing Ltd.

Kettunen, P., 2009. Adopting key lessons from agile manufacturing to agile software product development—A comparative study. Technovation 29, 408–422.

Kidd, P.T., 1996. Agile manufacturing: a strategy for the 21st century, in: IEE Colloquium on Agile Manufacturing. Presented at the IEE Colloquium on Agile Manufacturing, p. 1/1-1/6.

Kolodner, J., 1993. Case-Based Reasoning. Morgan Kaufmann.

L

Lechner, C., Floyd, S.W., 2007. Searching, Processing, Codifying and Practicing – Key Learning Activities in Exploratory Initiatives. Long Range Plann. 40, 9–29.

Liebowitz, J., 2001. Knowledge management and its link to artificial intelligence. Expert Syst. Appl. 20, 1–6.

Lin, C.-T., Chiu, H., Chu, P.-Y., 2006. Agility index in the supply chain. Int. J. Prod. Econ. 100, 285–299.

Linderman, K., 2003. Six Sigma: a goal-theoretic perspective. J. Oper. Manag. 21, 193–203.

Lindstrom, L., Jeffries, R., 2004. Extremme Programing and Agile Software Development Methodologies, Information Systems Management.

M

Malakooti, B., 2012. Decision making process: typology, intelligence, and optimization. J. Intell. Manuf. 23, 733–746.

Manifesto for Agile Software Development, 2001. URL https://agilemanifesto.org/

McCauley, R., 2001. Agile development methods poised to upset status quo. SIGCSE Bull. 33, 14–15.

McGuinness, D.L., 2002. Ontologies come of age. Spinn. Semantic Web Bringing World Wide Web Its Full Potential 171–194.

Minor, M., Bergmann, R., Görg, S., 2014. Case-based adaptation of workflows. Inf. Syst. 40, 142–152.

N

Nagel, R.N., Dove, R., 1991. 21st Century Manufacturing Enterprise Strategy: An Industry-Led View. DIANE Publishing.

Newell, A., Simon, H.A., 1972. Human problem solving. Prentice-Hall, Englewood Cliffs, NJ.

Nuseibeh, B., 2001. Weaving together requirements and architectures. Computer 34, 115–119.

P

Palmer, S.R., Felsing, J.M., 2002. A practical guide to feature-driven development. Upper Saddle River, N.J. ; [Great Britain] : Prentice Hall PTR.

Peral, J., Maté, A., Marco, M., 2017. Application of Data Mining techniques to identify relevant Key Performance Indicators. Comput. Stand. Interfaces 50, 55–64.

Prashar, A., 2017. Adopting PDCA (Plan-Do-Check-Act) cycle for energy optimization in energy-intensive SMEs. J. Clean. Prod. 145, 277–293.

Process Classification Framework | APQC, 2015. URL https://www.apqc.org/pcf

Q

Qumer, A., Henderson-Sellers, B., 2008. An evaluation of the degree of agility in six agile methods and its applicability for method engineering. Inf. Softw. Technol. 50, 280–

295.

R

Rakoto, H., 2004. Intégration du retour d'expérience dans les processus industriels : application à Alstom Transport (phd).

Raschke, R.L., 2010. Process-based view of agility: The value contribution of IT and the effects on process outcomes. Int. J. Account. Inf. Syst. 11, 297–313.

Raschke, R.L., David, J.S., 2009. Business process agility.

Ray, G., Barney, J.B., Muhanna, W.A., 2004. Capabilities, business processes, and competitive advantage: choosing the dependent variable in empirical tests of the resource-based view. Strateg. Manag. J. 25, 23–37.

Ren, J., Yusuf, Y.Y., Burns, N.D., 2009. A decision-support framework for agile enterprise partnering. Int. J. Adv. Manuf. Technol. 41, 180–192.

Riesenberger, C.A., Sousa, S.D., 2010. The 8D Methodology: An Effective Way to Reduce Recurrence of Customer Complaints? Lect. Notes Eng. Comput. Sci. 2185, 2225–2230.

Robey, D., Ross, J.W., Boudreau, M.-C., 2002. Learning to Implement Enterprise Systems: An Exploratory Study of the Dialectics of Change. J Manage Inf Syst 19, 17–46.

Romero Bejarano, J.C., 2013. Collaborative problem solving within supply chains: general framework, process and methodology. University of Toulouse.

Royce, W.W., 1987. Managing the Development of Large Software Systems: Concepts and Techniques, in: Proceedings of the 9th International Conference on Software Engineering, ICSE '87. IEEE Computer Society Press, Los Alamitos, CA, USA, pp. 328–338.

S

Sambamurthy, V., Bharadwaj, A., Grover, V., 2003. Shaping Agility through Digital Options: Reconceptualizing the Role of Information Technology in Contemporary Firms. MIS Q. 27, 237–263.

Sanchez, L.M., Nagi, R., 2001. A review of agile manufacturing systems. Int. J. Prod. Res. 39, 3561–3600.

Sangari, M.S., Razmi, J., Zolfaghari, S., 2015. Developing a practical evaluation framework for identifying critical factors to achieve supply chain agility. Measurement 62, 205–214.

Schreiber, G.T., Akkermans, H., 2000. Knowledge Engineering and Management: The CommonKADS Methodology. MIT Press, Cambridge, MA, USA.

Schwaber, K., 2004. Agile Project Management with Scrum. Microsoft Press.

Schwaber, K., Beedle, M., 2002. Agile Software Development with Scrum.

Schwaber, K., Sutherland, J., 2001. The Scrum Guide.

Seddon, P.B., Calvert, C., Yang, S., 2010. A Multi-project Model of Key Factors Affecting Organizational Benefits from Enterprise Systems. MIS Q 34, 305–328.

Seethamraju, R., Krishna Sundar, D., 2013. Influence of ERP systems on business process agility. IIMB Manag. Rev. 25, 137–149.

Seethamraju, R., Seethamraju, J., 2009. Entreprise system and Business Process Agility - A Case Study, in: Paper Presented at the 42nd Hawaii International Conference on System Sciences. pp. 1–12.

Sharifi, H., Zhang, Z., 1999. A methodology for achieving agility in manufacturing organisations: An introduction. Int. J. Prod. Econ. 62, 7–22.

Shewhart, W.A., 1930. Economic Quality Control of Manufactured Product1. Bell Syst. Tech. J. 9, 364–389.

Stapleton, J., 1997. DSDM, Dynamic Systems Development Method: The Method in Practice. Cambridge University Press.

Swenson, K.D., von Rosing, M., 2015. Phase 4: What Is Business Process Management?, in: The Complete Business Process Handbook. Elsevier, pp. 79–88.

Sylla, A., Vareilles, E., Coudert, T., Kirytopoulos, K., Aldanondo, M., Geneste, L., 2017. Readiness, feasibility and confidence: how to help bidders to better develop and assess their offers. Int. J. Prod. Res. 0, 1–19.

T

Tarhan, A., Yilmaz, S.G., 2014. Systematic analyses and comparison of development performance and product quality of Incremental Process and Agile Process. Inf. Softw. Technol., Performance in Software Development 56, 477–494.

Turban, E., Aronson, J.E., Liang, T.-P., 2004. Decision Support Systems and Intelligent Systems (7th Edition). Prentice-Hall, Inc., Upper Saddle River, NJ, USA.

V

Valle, S., Vázquez-Bustelo, D., 2009. Concurrent engineering performance: Incremental versus radical innovation. Int. J. Prod. Econ. 119, 136–148.

Van Rees, R., 2003. Clarity in the usage of the terms ontology, taxonomy and classification. CIB Rep. 284 432–9.

Vokurka, R.J., Fliedner, G., 1998. The journey toward agility. Ind. Manag. Data Syst. 98, 165–171.

von Rosing, M., Foldager, U., Hove, M., von Scheel, J., Falk Bøgebjerg, A., 2015a. Working with the Business Process Management (BPM) Life Cycle, in: The Complete Business Process Handbook. Morgan Kaufmann, Boston, pp. 265–341.

von Rosing, M., von Scheel, J., Gill, A.Q., 2015b. Applying Agile Principles to BPM, in: Scheel, M. von R.-W.S. von (Ed.), The Complete Business Process Handbook. Morgan Kaufmann, Boston, pp. 553–577.

von Scheel, H., von Rosing, M., Fonseca, M., Hove, M., Foldager, U., 2015. Phase 1: Process Concept Evolution, in: The Complete Business Process Handbook. Morgan Kaufmann, Boston, pp. 1–9.

W

Weber, B., Wild, W., 2005. Towards the agile management of business processes, in: Althoff, K.D., Dengel, A., Bergmann, R., Nick, M., RothBerghofer, T. (Eds.), Professional Knowledge Management. Springer-Verlag Berlin, Berlin, pp. 409–419.

Weber, B., Wild, W., 2004. An Agile Approach to Workflow Management, in: In Proceedings of Modellierung 2004.

Wieder, C., Blanco, E., Le Dain, M.-A., Trebucq, B., 2007. How to evaluate the NPD process agility in an intensive innovation context. Presented at the International Conference on Engineering Design (ICED'07).

Wu, Z., Palmer, M., 1994. Verbs Semantics and Lexical Selection, in: Proceedings of the 32Nd Annual Meeting on Association for Computational Linguistics, ACL ’94. Association for Computational Linguistics, Stroudsburg, PA, USA, pp. 133–138.

Y

Yager, R.R., 2004. Generalized OWA Aggregation Operators. Fuzzy Optim. Decis. Mak. 3, 93–107.

Yusuf, Y.Y., Sarhadi, M., Gunasekaran, A., 1999. Agile manufacturing:: The drivers, concepts and attributes. Int. J. Prod. Econ. 62, 33–43.

Z

Zorn, T.E., Taylor, J.R., 2004. Knowledge Management and/as Organizational Communication. Presented at the Key Issues in Organizational Communication, Routledge, pp. 96–99.

APPENDIX

Appendix 1

Manifesto for Agile Software Development *(“Manifesto for Agile Software Development,” 2001)*

We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:

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

That is, while there is value in the items on the right, we value the items on the left more.

Principles behind the Agile Manifesto *(“Manifesto for Agile Software Development,” 2001)*

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 for shorter periods of time.
- 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.

Appendix 2

Interview support: Identification and description of problem-solving processes

0 Analysis of the existing - Organization

1- Which are the standards used in the company for problem solving?
2- How do you store (and reuse?) problem-solving processes
3- How is in charge of making problem-solving related decisions (which process to follow, the composition of the team,…)?

1 Problem definition

Example of problems that have occurred at least 2 times (or similar problems) and their problem-solving process. Full description of the problem (at least):

1- Process model	e.g. 8D, PDCA, DMAIC …
2- Enterprise department	e.g. Quality, Manufacturing, HR …
3- Type of product	e.g. saddle, frame …
4- Type of problem	e.g. dimensional, painting …
5- Cost constraints	…
6- Delay constraints	…

Strict or negotiable constraints?

2 Resolution of the problem

1- Who are the people in charge of choosing the way to solve the problem? Who did solve it?

2- Which was the standard to follow? Was it respected?

3- How did the process evolved? Did any unexpected event occur?
How did constraints of cost and delay evolve?

4- Which was the company context? Did it influenced the problem and/or its resolution?

5- Please provide all information that helped during decision making

3 Your needs

1- Do you have identified needs regarding current problem-solving processes?
2- Do you think that a knowledge-driven system could improve your processes?
3- Do you think that an evaluation of the process KPIs on run-time could improve and/or help decision making?
4- Other needs …?

4 Process indicators

1- How long does it take you to solve problems with the NC method? And with the 8D?

2- How many resources are involved in problem solving for each method?

ABSTRACT

In order to survive to the unstable and highly changing market-place, modern organisations need to adapt their business processes to be more agile. Such is, particularly, the case of problem solving processes. Problem solving is a key activity that companies perform on a daily basis to improve quality and to obtain sustainable and continuous improvement. Such processes are built following standard rigid frameworks as Plan, Do, Check, Act (PDCA), Define, Measure, Analyse, Improve, Control (DMAIC), or 8 Disciplines (8D)/ 9 Steps (9S). In these methods, the generalization and reuse of knowledge is facilitated by standardization. However, it is sometimes difficult to react to unexpected events due to over-constrained standards. Then, a need arises to define a problem solving process sufficiently structured but not over constrained by standards, which can be reconfigured and adapted to unexpected situations, and that is based on experience feedback principles.

This thesis work describes a proposition of an agile problem solving process driven by the reuse of experiences and knowledge. For this purpose, based on Case-Based Reasoning (CBR) principles, the complete lifecycle of an agile problem solving process is proposed. Following the five steps that compose the agile lifecycle, the agile process can be defined, executed and stored in a dedicated knowledge and experience base. An application of the model to a specific problem solving process of a surface treatment company is presented. The process is analysed, deploying the complete agile lifecycle. It is shown how the standard problem solving method used within the company could become more agile through the application of our method.

Keywords: Problem Solving, Agility, Business Process Management, Experience Feedback, Knowledge Management.

RESUME

Les organisations d’aujourd’hui ont besoin d’être plus agiles afin de survivre dans des marchés fluctuants et instables. C’est le cas particulier des processus de résolution de problèmes. La résolution de problèmes est une activité clé que les entreprises réalisent quotidiennement afin d’améliorer leur qualité et de réussir l’amélioration continue globale. Ces processus sont construits à partir des standards cadrés tels que le Plan, Do, Check, Act (PDCA), Define, Measure, Analyse, Improve, Control (DMAIC), ou le 8 Disciplines (8D)/ 9 Steps (9S). Dans ces méthodes, la généralisation et la réutilisation des connaissances sont facilitées par la standardisation. Cependant, les standards ayant tendance à contraindre fortement les processus, il est parfois difficile de réagir face à des évènements imprévus ou même de s’écarter pour mieux répondre aux besoins. Ainsi, le besoin de processus de résolution de problèmes suffisamment structurés mais pas sur-contraints par des standards apparaît. Un tel processus doit pouvoir être reconfiguré et adapté à des situations inattendues et se baser sur des méthodes de retour d’expérience.

Cette thèse décrit la proposition d’un processus agile de résolution de problèmes guidé par le retour d’expériences et les connaissances. A cet effet, le cycle de vie d’un processus agile de résolution de problèmes, basé sur les principes du Case-Based Reasoning (CBR), est proposé. Au travers des cinq étapes d’un cycle de vie agile, le processus peut être défini, réalisé et stocké dans des bases d’expériences et de connaissances spécifiques à des fins de réutilisation. L’application du modèle à un processus de résolution de problèmes dans une entreprise de traitement de surface est présentée. Le processus est analysé en déployant le cycle de vie agile. Il est montré comment la méthode standard de résolution de problèmes utilisée au sein de l’entreprise peut devenir plus agile grâce à l’application de notre méthode.

Mots clés: Résolution de problèmes, Agilité, Gestion de processus d’entreprise, Retour d’expérience, Gestion des connaissances.