Università
Ca'Foscari
Venezia

Master's Degree programme –
Second Cycle (*D.M. 270/2004*)
In Economia e Gestione delle Aziende
Curriculum International Management

Master's Degree Thesis

Ca' Foscari
Dorsoduro 3246
30123 Venezia

THE EVOLUTION OF PROJECT MANAGEMENT

Supervisor
Ch. Prof. Anna Comacchio
Assistant supervisor
Ch. Prof. Anna Cabigiosu

Graduand
Veronica Bortolussi
829974

Academic Year
2015 / 2016

Table of Contents

Acknowledgment

I would like to express my sincere gratitude to my advisor, Professor Anna Comacchio. Her support and enthusiasm over the course of my dissertation was invaluable. Mrs. Comacchio's expertise and willingness to help, provided me with the support I needed to develop my dissertation, while giving me the freedom to work in my own way.

I would also like to thank my boss and colleague, Fabio, who introduced me to the Agile world. Without his in depth advice and knowledge, I am sure my paper would have turned out much differently.

Thank-you to my university friends. You have been great study partners and an extraordinary support system.

Finally, I want to express my profound gratitude to my family. Thank you to all those who have been there for me during my studies. In particular, to my mother; thank you for giving me the opportunity to study, and for supporting me both emotionally and financially over the past years. Finally, thank-you to those who are not here anymore. Although your physical presence is not here, you have never ceased to inspire me.

INTRODUCTION

In order to stay ahead in today's competitive and chaotic global economy, companies are increasingly focusing on project management in order to consistently deliver strong business results. Project management stands for: project, program, and portfolio management. Companies are becoming more aware of the payoffs they can obtain from investing time, money, and resources, to build organizational project management expertise, such as lower costs, greater efficiencies, improved customer and stakeholder satisfaction, and greater competitive advantage. Disciplined project management begins with the portfolio, where the company's strategic vision drives initial investments, and value measures are established.

A fully aligned project management strategy encompasses the entire organization, it drives project execution at every level, and aims to deliver value at each step during the timeline of the project. Meanwhile, a strong organization-wide commitment to project management leads to better results across multiple projects and years, and leads to long-term business value.[1]

Adrian McKnight, program director at Suncorp-Metway Ltd. (a financial services firm in Brisbane, Australia), states that: "The delivery of business outcomes is realized through the success of projects, and in essence that is the way that project management strategies drive organizational success". A survey by consulting company McKinsey & Co.[2] found that nearly 60 percent of senior executives agreed building a strong project management discipline to be a top three business priority as they plan for the future.

The importance and relevance of project management strategies and discipline for companies is also supported by project-based organizations. The current types of project management found in organizations can be divided into three categories: functional, matrix, and project-based.

The functional organization works inside the organization hierarchical structure. It ensures that the various components of the project are undertaken by each functional unit, and that each

[1] Kodama M., (2007). "Project-Based Organization in the Knowledge-Based Society", London, Uk: Imperial College Press.
[2] McKinsey & Co., January 2010. Results based on a survey of 1,440 senior executives.

unit is responsible for its charged component. Therefore, the functional component plays a dominant role in project management.

The matrix structure is a hybrid. Each level of the project management structure is loaded on the functional hierarchical structure, and this lead to a higher dynamism of the organization. Different types of matrix systems exist in practice, depending on the balance between the control and authority exerted by the project manager and the functional manager on the project.

In contrast to the above-mentioned forms, the project-based organization is one in which the functional organization has become obsolete. Without the formal functional coordination of activities, the project is the primary business mechanism for coordinating and integrating all the main business functions of the firm, primarily production, engineering, Research and Development, personnel, and marketing. This type of organization is entirely dedicated to multiple projects. An independent project team is created within the organization, the team's management is separated from the other units, and from the parent organization (if present). It has its own staff and management, where the project manager is given direct control over business functions, personnel, and other resources specifically allocated to the project team by the enterprise.

Essentially, project-based organizations are the types of organizational forms that are optimized to achieve specific work targets. They formulate and implement future strategies through the integration of resources and capabilities from outside organizations.

"Project-based organizations refer to a variety of organizational forms that involve the creation of temporary systems for the performance of project tasks"[3]. This definition clarifies that project-based organizations incorporate a specially formed structure for a temporary period of time, to enable the execution of a specific task. This temporary organizational form is the key characteristic of project-based organizations. They are flexible and reconfigurable, in contrast to the traditional, large integrated, hierarchical organizations that tend to be less innovative.[4]

[3] Sydow R., Lindkvist L., DeFillippi R., (2004). "Project-Based Organizations, Embeddedness and Repositories of Knowledge: Editorial", SAGE publications. P.1475.
[4] Thiry M., (2006). "Beyond the Matrix: The Integrated Project-Based Organization", PMI Global Congress Proceedings.

Within a pure, project-based organization (an organization in which no other form is present), major projects will embody the majority of the business functions that are normally carried out by the departments with the functional and matrix structure. Project-based organizations arrange their structure, strategy, and capabilities around the needs of their projects, which often cut across conventional boundaries.[5] In some cases, the project involves a consortium of companies (e.g., Sematech, Airbus, the Channel Tunnel, and the Millennium Dome)[6]. In others, much or all of the project may be carried out within the boundaries of a single company.

The structures and business processes in these kind of organizations are likely to be shaped by the changing profiles of their projects, in particular, by their size, complexity, and duration. At the same time, the ability to adapt like this is a strength. The project-based organization with its knowledge, capabilities, and resources, are built up through the execution of major projects.

This explanation has provided an outline of the importance of project-based organizations, and of the way they are managed inside of the firm. To sum up, as explained above, a project is a type of organizational arrangement that often has an agreed and predefined time frame for initiation and completion among the parties involved. It is usually task-driven as it brings together and coordinates the specialized skills, resources, and knowledge required to complete the project. This is done within the predefined time, cost, quality, and other business constraints, typically by carrying out a series of non-routine activities.

In this scenario, the importance of project management is clear: it plays a leading role in managing risks, quality, changes and integration with business processes, and the systems within organizations. Project management also has an essential role in identifying and resolving issues, and in capturing and managing project knowledge. Lastly, it ensures that key lessons are learned and carried forward from the success or failure of the projects, which is the key to business innovation and quality improvement.

This dissertation aims to give the reader an overview of the evolution of project management, from the point of view of the techniques that have changed, improved and emerged over time,

[5] Cattani G., Ferriani S., Frederiksen L., Täube F., (2011). "Project-Based Organizing and Strategic Management", Emerald Group Publishing Limited.
[6] Hobday M., (2000). "The project-based organisation: an ideal form for managing complex products and systems?", Brighton, UK: Research Policy 29. P. 874.

while taking into account the different aspects and philosophies of the project management discipline.

Although its term is recent, it is not difficult to realise that project management has always existed; just think of the massive construction projects undertaken in the ancient worlds, for example ancient Egypt, Greece, and Rome. It was only few thousand years forward in time, at the beginning of the twentieth century, when Frederick Taylor (1856-1915) and Henry Laurence Gantt (1861-1919), began to formulate theories and ways to optimize processes to achieve specific organizational objectives.

It took until the early 60s before project management began to be considered a real discipline. Since then, it has increasingly become more and more important. Time and development brought a series of new theories, methodologies, and standards. These aimed to improve the systems management, business strategies, organizational changes, and interpersonal relations. Nevertheless, it took until the early 2000s until the role of a project manager was recognized as a crucial entity for the successful achievement of a project, giving rise to a massive increase of interest and attention to the subject. This dissertation will focus on the most recent period of project management history.

CHAPTER 1. TRADITIONAL PROJECT MANAGEMENT

1.1 Basic Concepts of Project Management

The first part of this chapter is developed on the basis of the work of a particular organization, the Project Management Institute (PMI). Founded in 1969 with the purpose of setting standards for project management, PMI studies how to improve the way projects are managed and provides the increasing number of project managers the opportunity to share knowledge and learn more about this discipline. PMI is now the world's leading not-for-profit professional membership association for the project management profession, and is recognized by the American National Standards Institute (ANSI) as an accredited standard developer. One standard in particular is the Guide to the Project Management Book of Knowledge (PMBOK Guide): born in 1987 as an attempt to define and standardize the information and practices of project management, it has been generally accepted by the community of project managers[7].

The second part of the chapter aims at giving a brief but as clear as possible explanation of the techniques used in traditional project management, in order to put the basis for the subjects discussed in the next chapters.

1.1.1 Definitions of Project and Project Management

A project is "a temporary endeavor undertaken to create a unique product, service, or result"[8]. The use of two important words in PMI's definition of project management must be underlined: temporary and unique.

A project is temporary because it necessarily has definite beginning and end in time, and consequently defined extent (scope) and resources.

A project is unique because activities and operations used to accomplish a project are specific and not routine. In the same way, a project team often includes people who usually do not

[7] Lawson D., (2009). "PMBOK Quick Implementation Guide", Emereo Publishing.
[8] Project Management Institute, (2013). "A guide to the Project Management Body of Knowledge – Fifth Edition (PMBOK Guide)". Newtown Square, Pennsylvania: Project Management Institute. P. 3.

work together and that have been grouped together thanks to differences in specific competences, skills or geographies.

Then the question arises: what does *project management* mean? Project management must be intended as "the application of processes, methods, knowledge, skills and experience to achieve the project objectives. […] A project is usually deemed to be a success if it achieves the objectives according to their acceptance criteria, within an agreed timescale and budget." [9]

To establish a project management business system it is necessary to intend that system as a temporary combination of people, resources, organizational factors and tools to reach objectives those are unique, defined and bounded in time, costs, quality and limited resources.

The concept of organizational factors must be underlined because companies should invest and make an effort on it, in defining and developing a good project management system, since people, tools and techniques quickly change and evolve while organizational tools remain and guarantee the continuity of the system itself. The concept of barrier also deserves a highlight, as are these boundaries, more and more growing and pressing for the companies, that cause more prudent and rational project management to be necessary. When barriers imposed by the market, customers, or norms about project management will make increasingly compulsory the respect of contractual commitment on defined schedules, costs and quality of projects, then an adequate project management system will be mandatory for companies.

Going back to the point, project management is "the application of knowledge, skills, tools, and techniques to project activities to meet the project requirements."[10] Thus, managing a project during its entire lifecycle means appropriately applicate and integrate a sequence of strongly related management processes. The typical phases to run in managing a project are:

- Identifying requirements;
- Addressing the various needs, concerns and expectations of the stakeholders as the project is planned and carried out;
- Setting and maintaining active communication with stakeholders;

[9] APM – Association for Project Management

[10] Project Management Institute, (2013). "A guide to the Project Management Body of Knowledge – Fifth Edition (PMBOK Guide)". Newtown Square, Pennsylvania: Project Management Institute. P. 47.

- Balancing the competing project constraints, which includes: Scope, Quality, Schedule, Budget, Resources and Risks.

The specific project circumstances will influence the constraints on which the project manager needs to focus and require effective application and management of appropriate project management processes.

1.1.2 Project Management processes

A process is defined by PMI as "a set of interrelated actions and activities performed to create a pre-specified product, service, or result. Each process is characterized by its inputs, the tools and techniques that can be applied, and the resulting outputs"[11].

The PMBOK Guide identifies 47 project management processes and groups them into five categories known as Project Management Process Groups: Initiating, Planning, Executing, Monitoring and Controlling, Closing. Below is a summary explanation of which processes and procedures should be conducted by the organization for each project phase[12].

Initiating

The goal of this phase is to define the project at a broad level. Depending on the nature of the project, feasibility studies are conducted. For an IT project in this phase requirement gathering and analysis are performed. Once a project obtains the green light from stakeholders, it is needed to create documents outlining the purpose and requirements of the project.

Planning

This phase focuses on creating a roadmap for the entire project: here are the processes through which are defined project objectives and are identified and selected the best paths to reach them.

Some of the important activities included in this phase are: scope planning and definition, definition and estimation of activities needed to complete deliverables, making WBS (Work

[11] Project Management Institute, (2013). "A guide to the Project Management Body of Knowledge – Fifth Edition (PMBOK Guide)". Newtown Square, Pennsylvania: Project Management Institute. P. 47.
[12] Weiss J. W., Wysocki R. K., (1992). "Five-Phase Project Management: A Practical Planning and Implementation Guide", Perseus Books.

Breakdown Structure), development of schedule, milestone charts, GANTT charts, estimating and allocating resources, establishment of baselines or performance measures, planning dates and modes of communication with stakeholders based on milestones, deadlines and important deliveries. An important part of planning phase is risk management planning, which includes risk identification and analysis, risk mitigation approaches and risk response planning.

Executing

In this phase are the processes aimed to coordinate the resources assigned to the project in order to accomplish its objectives in adherence to the plans.
Executing processes include also other activities not strictly related to project execution, namely quality assurance, project development updates, human resource development, performance reports, status reports, and others.

Controlling

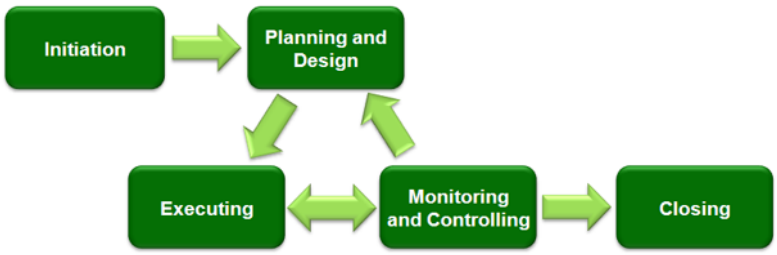
This phase is focused on those processes aimed to monitor and measure the project performance and progression in order to ensure the achievement of project objectives and adherence to plans. Controlling processes includes: scope verification and control (to check and monitor for scope advancement), change control (to identify and manage changes to project requirements), calculating key performance indicators for cost and time to detect any variation from what scheduled.
All these processes are to measure the extent of variation from the project management plan and in case corrective measures are taken or adjustments to project plan are generated repeating respective planning processes. This group of processes rests on a fundamental basis of project management: control can exist only if a reference plan exist.

Closing

This phase is aimed to formally close a project. Closing processes include important tasks such as making the delivery, relieving resources, reward team members, and formal termination of any eventual contractor involved in the project.

Here on the side is an operative flow chart of these five macro-processes and interactions (Figure 1).

Figure 1 - Macro processes and interactions, operative flow chart



Source: processglobal.biz

Except the two processes of initiating and closing, it is clear how project management is a continuous cyclic reiteration of three processes of planning, executing and controlling. The 47 project management processes identified in the PMBOK Guide are grouped also into ten separate Knowledge Areas.

A Knowledge Area represents an area of specialization, a professional field, made of a complete set of concepts, terms, activities and tasks. These ten Knowledge Areas are used on most project most of the time, but at the same time it is clear that project teams should utilize them properly for their specific project.[13]

The Knowledge Areas are: Project Integration Management, Project Scope Management, Project Time Management, Project Quality Management, Project Human Resource Management, Project Communications Management, Project Risk Management, Project Procurement Management and Project Stakeholder Management. Each of the ten knowledge areas contains the processes that need to be accomplished within its discipline in order to achieve effective project management. Each of these processes also falls into one of the five process groups, creating a matrix structure such that every process can be related to one knowledge area and one process group (Figure 2).

[13] Wysocki R., (2013). "Effective Project Management: Traditional, Agile, Extreme", Wiley.

Figure 2 - Matrix structure showing how every process is related to one knowledge area and one process group.

Knowledge Areas	Project Management Process Groups				
	Initiating	**Planning**	**Executing**	**Monitoring & Controlling**	**Closing**
Project Integration Management	4.1 Develop Project Charter	4.2 Develop Project Management Plan	4.3 Direct and Manage Project Work	4.4 Monitor and Control Project Work 4.5 Perform Integrated Change Control	4.6 Close Project or Phase
Project Scope Management		5.1 Plan Scope Management 5.2 Collect Requirements 5.3 Define Scope 5.4 Create WBS		5.5 Validate Scope 5.6 Control Scope	
Project Time Management		6.1 Plan Schedule Management 6.2 Define Activities 6.3 Sequence Activities 6.4 Estimate Activity Resources 6.5 Estimate Activity Durations 6.6 Develop Schedule		6.7 Control Schedule	
Project Cost Management		7.1 Plan Cost Management 7.2 Estimate Costs 7.3 Determine Budget		7.4 Control Costs	
Project Quality Management		8.1 Plan Quality Management	8.2 Perform Quality Assurance	8.3 Control Quality	
Project Human Resource Management		9.1 Plan Human Resource Management	9.2 Acquire Project Team 9.3 Develop Project Team 9.4 Manage Project Team		
Project Communications Management		10.1 Plan Communications Management	10.2 Manage Communications	10.3 Control Communications	
Project Risk Management		11.1 Plan Risk Management 11.2 Identify Risks 11.3 Perform Qualitative Risk Analysis 11.4 Perform Quantitative Risk Analysis 11.5 Plan Risk Responses		11.6 Control Risks	
Project Procurement Management		12.1 Plan Procurement Management	12.2 Conduct Procurements	12.3 Control Procurements	12.4 Close Procurements
Project Stakeholder Management	13.1 Identify Stakeholders	13.2 Plan Stakeholder Management	13.3 Manage Stakeholder Engagement	13.4 Control Stakeholder Engagement	

Source: PMBOK Guide, 2013.

1.2 Traditional Project Management

Focusing on the planning process, the first question that comes to mind is 'what to do?'. Traditional project management provides some techniques in response to that question: the Work Breakdown Structure, the Gantt chart and the PERT methodology. An explanation of each one is given below.

1.2.1 The Work Breakdown Structure[14]

The PMBOK Guide defines a Work Breakdown Structure as "a deliverable-oriented hierarchical decomposition of the work to be executed by the project team to accomplish the project objectives and create the required deliverables. It organizes and defines the total scope of the project. Each descending level represents an increasingly detailed definition of the project work. The WBS is decomposed into Work Packages."[15].

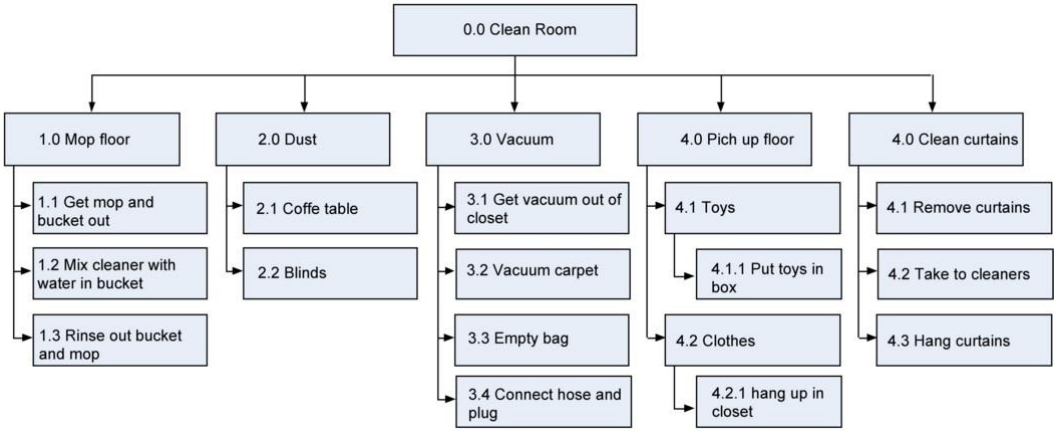
For *deliverable* is intended any outcome, result or item that can be measurable, tangible or verifiable that must be produced to complete a project or part of a project.

Work packages are defined by PMI as the lowest level WBS components, where planned work is contained, and for which cost and duration can be estimated and managed. The level of detail for work packages will vary with the size and complexity of the project and the level of decomposition often vary with the degree of control needed to effectively manage the project. In practice, a work package consists of one continuous activity, for example, the use of a single piece of equipment, until the work package is complete. A successful work package has a definite beginning and ending tasks, so that its completion will be easily measured and clearly observed: one of the most important part of the work package is the description of each of the tasks that comprise it. Each task (and consequently the work package) is considered complete when the deliverable has been provided.[16]

[14] Norman E. S., Brotherton S. A., and Fried R. T., (2011). "Work Breakdown Structures: The Foundation for Project Management Excellence", Wiley.
[15] Project Management Institute, (2013). Glossary of "A guide to the Project Management Body of Knowledge – Fifth Edition (PMBOK Guide)". Newtown Square, Pennsylvania: Project Management Institute.
[16] Golany B., Shtub A., (2001). "Handbook of Industrial Engineering: Technology and Operations Management", John Wiley & Sons.

In other words, WBS represents a tool to operate a subdivision of project deliverables and project work, usually very complex, into smaller and more manageable components, as long as an adequate degree of granularity is reached. In the end, there is a detailed framework of the categories of work and the necessary activities involved in completing the project, that can be presented in either an outline format or a diagram, a graphical hierarchy or interlocking tiers and boxes (Figure 3).

Figure 3 - Example of Work Breakdown Structure for cleaning a room



Source: www.opentextbooks.org.hk

The key benefit of this process is that it provides a structured vision of what has to be delivered, and this brings other benefits: team members understand the deliverables of a project, conversation and interaction between them begins, they see how their contribution benefits the overall project, and last but not least team discussion can mitigate the risk of delays or missing work.[17]

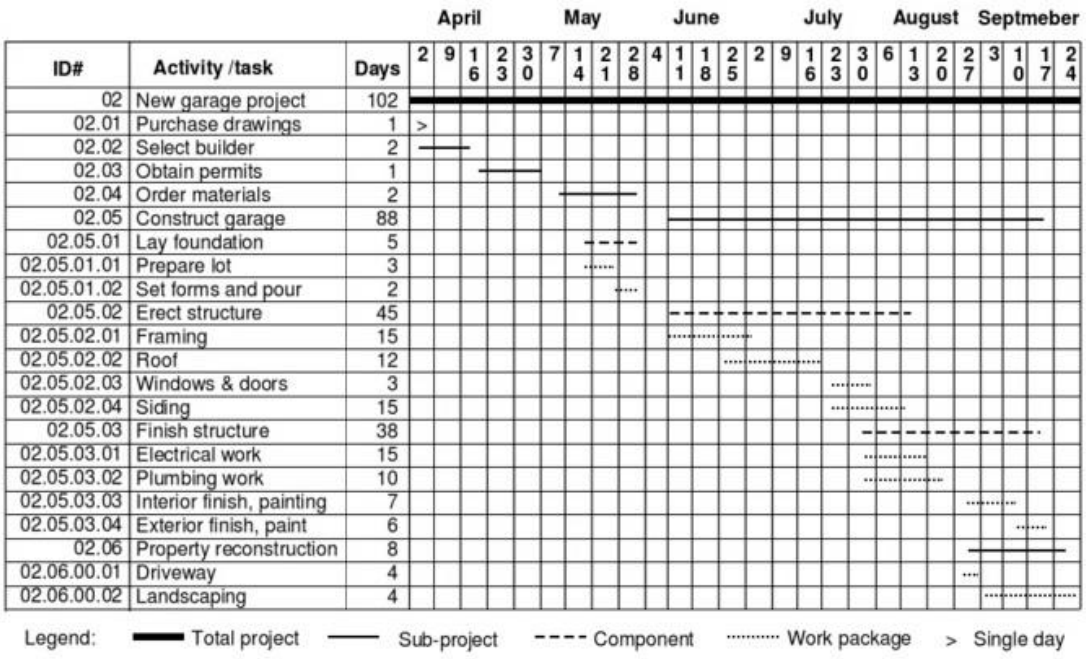
[17] Brotherton S.A., Fried R.T., Norman E.S., (2008). "Applying the Work Breakdown Structure to the Project Management Lifecycle", PMI Global Congress Proceedings, Denver, Colorado.

1.2.2 The Gantt chart

The Gantt chart, invented in 1917, is one of the most used, convenient, and clear depictions of project activities. It is a two-dimensional graphical representation of the activities in the project: the vertical dimension lists the project activities, while the horizontal dimension represents time.[18]

The Gantt chart is used both as a planning document and as a way to document actual performance. Major project work packages or task clusters are listed vertically on a time grid so as to depict the planned start and finish times and activity duration as a bar across the chosen time intervals (weeks, months, quarters). Columns can be added to show the dates of planned start and finish for each timeline. Major milestones, those are critical review and decision points, can be shown with a symbol along the major timelines. Here is an example of a Gantt chart for a garage building project (Figure 4).

Figure 4 - Example of Gantt chart for a garage building project.



Source: Westcott, 2004

The Gantt chart can be used also as a recording and reporting tool for documenting actual performance as the project progresses. To do this, an additional line below each planned

[18] Westcott R. T., (2004). "Simplified Project Management for the Quality Professional: How to Plan for and Manage Small and Medium-Size Projects", ASQ Quality Press.

timeline can be used to record actual dates, times, costs, and a bar is added to show the actual task duration.

1.2.3 The PERT Methodology and the Critical Path Method

Highly complex projects need a tool that would display the interdependencies of each of the project's tasks. Gantt chart can not show this kind of relationship because it is based on a time line[19]. For this reason, the next step is going to analyze other project-management methods useful for the purpose: PERT and Critical Path Method[20]. Although they are two different techniques, the two methods are similar enough to let many authors[21] feel allowed to analyse them together and consider as complementary, as both are directed to optimize the realization time of a project. In this work it has been made the same choice, as this section is just an introductive explanation of traditional project management. Nevertheless, the main differences between the two methods will be highlighted in the end of this paragraph.

PERT, acronym that stands for Program Evaluation Review Technique, is a methodology originated in the engineering profession in the late 1950s. It is one of the first methodologies that truly understood and considered the uncertainties that a project can present, and create a step-by-step process to help the Project Manager in understanding how to successfully deliver the project despite these uncertainties. PERT chart is based on a flow-chart concept and can demonstrate the critical relationship between any of the tasks in a project and help a project manager develop sophisticated calculation about the impact of task delays on final completion dates.

PERT consists of 5 steps:

- Identify activities and milestones, and consequently create a Work Breakdown Structure;
- Determine activity sequence;
- Create network diagram;

[19] Taylor J. C., Lashman R., Helling P., (1994). "Practical Problem-Solving Skills in the Workplace", Amacom.
[20] Morris R. A., McWhorter S. B., (2008). "Project Management That Works: Real-World Advice on Communicating, Problem-Solving, and Everything Else You Need to Know to Get the Job Done", Amacom Books.
[21] E.g. Morris, Rick A., Sember, Brette McWhorter.

- Estimate time;
- Determine critical path.

Observing this sequence of operations it is clear that the goal of PERT is to identify the *critical path* of a project. The definition for critical path is "the longest path through which represents the shortest amount of time in a project."[22]

A project team identifies tasks, time needed to complete each task and dependencies between tasks, below in Table 1 is given an example:

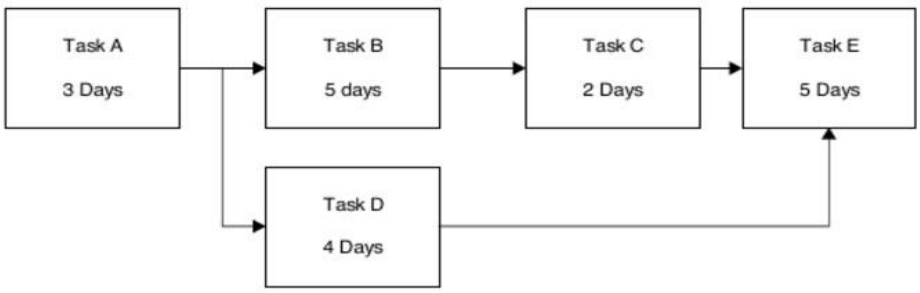
Table 1 – Example of tasks, time needed and dependencies between tasks.

Task	Time to complete	Dependency
A	3 days	None
B	5 days	Task A must finish
C	2 days	Task B must finish
D	4 days	Task A must finish
E	5 days	Task C and D must finish

Source: Morris, 2008

Based on the information in the table, an Activity Network Diagram (AND) can be created to depict the interrelationships of each task in the project (Figure 5). Once the activities or tasks have been sequenced, the Critical Path Method chart adds the dimension of a single time estimate to complete tasks.

Figure 5 - Example of Activity Network Diagram



Source: Morris, 2008

[22] Morris, Rick A., Sember, Brette McWhorter, (2008). "Project Management That Works: Real-World Advice on Communicating, Problem-Solving, and Everything Else You Need to Know to Get the Job Done". Amacom Books. P. 76.

Now the critical path can be determined. To do it, follow each path of the network diagram and determine how long each path will take. In the given example, two paths can be identified: one is A-B-C-E which will take 15 days, and the other is A-D-E which will take 12 days. The critical path for this diagram is A-B-C-E, because it is the longest one in terms of time needed to complete it. Identifying the critical path allows the project manager to take strategic decisions. For example, in the tasks previously considered, if the project manager needs to complete the project one day early, which would he/she choose between shortening Task B or Task D? The correct answer is Task B, because it is on the critical path. Choosing to shorten Task D would shorten that specific task, but would not affect the critical path, because the time needed to complete the project would not change, since there is nothing else depending on Task D. Otherwise, shortening Task B (in this specific scenario) will allow other dependent tasks to start earlier, and consequently to shorten the total duration of the project.

It can be inferred that "the critical path method identifies the tasks that a project manager should focus on to reduce the overall time frame of the project. In addition, the critical path tasks hold the highest risk. If a critical path task is late, the project date will move with it."[23]

As anticipated at the beginning of this paragraph, the most important differences between PERT and CPM are here provided.

The primary point that distinguishes PERT from CPM is that the first gives the maximum importance to time: if the time is minimized, consequently the cost will also be reduced. On the other side, the second takes as basic element the cost optimization, but considers time as well.

Secondly, PERT is a project management technique used to manage uncertain activities of a project, and is set according to events, while CPM is a statistical technique of project management that manages well defined activities of a project, and is set according to activities.

Another difference is that as PERT uses three time estimates for each work package, i.e. optimistic time, most likely time, and pessimistic time, it is best suited for high precision time

[23] Morris, Rick A., Sember, Brette McWhorter, (2008). "Project Management That Works: Real-World Advice on Communicating, Problem-Solving, and Everything Else You Need to Know to Get the Job Done". AMACOM Books. P. 77

estimates. Unlikely, CPM uses only one estimate and it is more suitable for a reasonable time estimate.

Related to this concept, it can be stated also that PERT is most often used for projects for which there may be little or no prior experience, while CPM is used where data are available for reasonable accurate time estimates.

Last but not least, PERT is more often used for projects where the nature of the job is non-repetitive and activities tend to be unpredictable, while CPM involves the job of repetitive nature and activities tend to be predictable. That is why PERT is preferred for R&D projects and CPM is preferred for non-research projects like construction projects.

To conclude, even if the differences between PERT and CPM are undeniable and deep in some case, there is evidence that the two methods are used in the successful completion of a project and hence used in conjunction with each other. That is the reason why the difference between these two project management tools is getting blurred: as time went on the techniques merged and in many projects are being used as a single technique.

1.3 The failure of traditional project management

Using the example of a project developed by external suppliers, customers who pay another company for carrying out a project feel uncomfortable and nervous about waiting to get it. What if they pay big money and find out just at the very end of the project that what the supplier (or the service provider) delivered is not what they wanted or needed?[24] This is a tangible issue for projects run with the traditional project management methods.

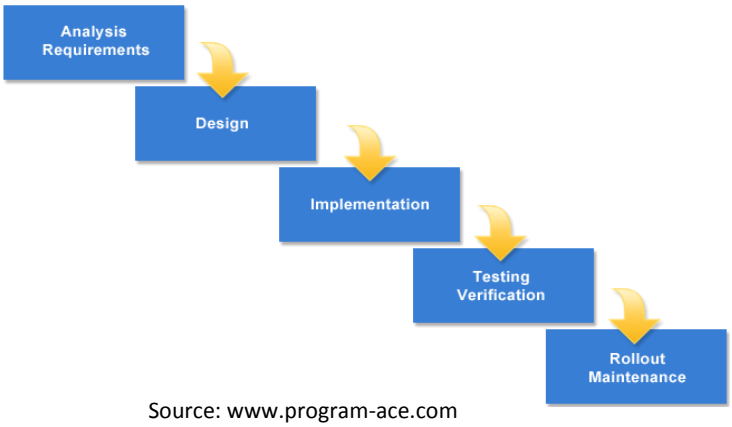
1.3.1 Waterfall

The traditional way of doing project management is called *waterfall*, because the phases of the project (requirements, functional specifications and design) are like steps in a waterfall: once one stage is finished, you pass to the next stage and can not go backward (Figure 6). This kind

[24] Wrubel E., Gross J., (2015). "Contracting for Agile Software Development in the Department of Defense: An Introduction", Software Engineering Institute, Carnegie Mellon University.

of development is so called because it is entirely planned in advance with no response to changes throughout the life of the project: flexibility is not part of waterfall project management.

Figure 6 - Waterfall project methodology. At the end of each stage, the project spills down to the next level. It is not possible going backwards but restarting.



Source: www.program-ace.com

So, talking about a big project it can happen that by the time it is completed, it might not even have resulted in what the customer wanted, as there is not a lot of communication within customer and supplier after the initial planning phase is finished.

Another issue that usually occur in waterfall projects is that it is never really made a diagnosis on what went wrong in order to "learn the lesson", and this makes it more likely to repeat similar mistakes in future projects.

1.3.2 Why do projects fail?

Many projects fail because they are carried out with the traditional way by people or companies who refuse to pursue innovation. It should be remembered that the basic project management principles outlined above in the chapter were developed in the 19th century.

The failure of traditional project management is a single problem that may reflect two different issues: a behavioural and psychological limit on one hand, and a structural limit of the methodologies on the other hand.

Concerning the behavioural and psychological limit, it is a fact that many project managers are unable to stop sticking to established methods for many reasons.[25]

One first reason to bring as an example is the belief that the processes can not be changed because that is the way management has decreed it. Actually, this is not the case, as the

[25] Wysocki R., (2013). "Effective Project Management: Traditional, Agile, Extreme", Wiley.

processes have evolved and redesigned. Consequent to this is the belief that processes are there for a reason, so trying to change them will lead to chaos.

Another reason is the natural resistance to change, intended as reluctance to move out of the comfort zone, the feeling that it is all too difficult, simple fear of the unknown. This often leads to the feeling that it is someone else's responsibility to make changes, especially prevalent in an authoritarian organisation, where methodologies are used as a substitute for thought.[26]

About the structural limit of traditional project management, the issue concerns right the concept of waterfall, the way of carrying out activities until the completion of the project. Some concrete issues are set out below.

Firstly, as the project is released entirely at its completion, since no incremental releases are in the plan, no value is gained from the project until it is finished.

In the same way, testing activities are not carried out until the later stages of the project: the eventual uncovering of big problems can represent a serious issue, because it might be too late to fix them.

Thirdly, customers see and are expected to approve the project once it is concluded, but in the meantime their requirements might have changed.

Even if customers communicate a requirements variation, no change can be made to plans due to inflexibility of waterfall methodologies.

Last but not least, another issue to consider is the tendency of waterfall methodology to rely on a single person direction: an eventual quit or replacement of the project manager may put the project in danger.

[26] Whitaker K., (2009). "Principles of Software Development Leadership: Applying Project Management Principles to Agile Software Development", Cengage Learning.

1.4 A better way to manage new product development projects: Stage-Gate Process

As the years passed by, the market increased its pressure on companies, which began to face an increasing national and international competition, more and more demanding customers, and a consequent need for shorter product cycle-time. The need for an effective project management to go along with the changing world and gaining a competitive advantage rapidly emerged.

During the 1980s, in response to the downsides of traditional project management, the pioneer of NPD (New Product Development) research Robert G. Cooper developed and proposed a new tool to help companies in facing the increasing market pressure: Stage-Gate.

"A Stage-Gate system is both a conceptual and an operational model for moving a new product from idea to launch"[27], in other words it is a road map for new product development process in order to improve efficiency and effectiveness.

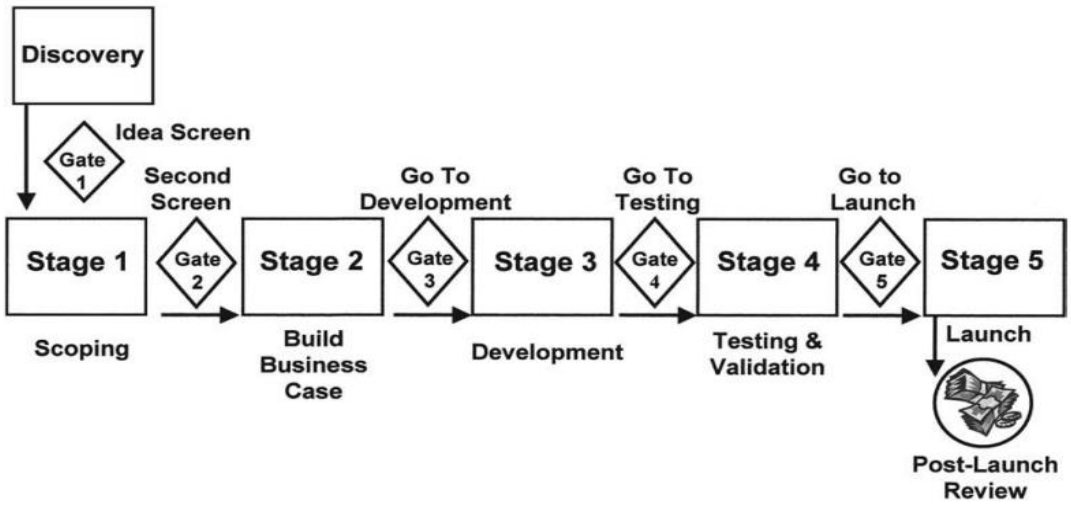
1.4.1 How does Stage-Gate works

As the name imply, Stage-Gate divides the dynamic process of product innovation in a series of stages, each one followed by a gate. Stages are where the project team carries out activities and the work is done, while the gates are where it is decided whether continue to invest in the project or abort it on the basis of its quality.

The original formulation of this model provides for five stages after the discovery (Figure 7):[28]

[27] Cooper R.G., (1990). "Stage-gate systems: A new tool for managing new products", Business Horizons. P. 44.
[28] Product Development Institute, (2016). "Stage-Gate – Your roadmap for New Product Development".

Figure 7 - An overview of a typical Stage-Gate System



Source: Cooper, 2001.

Stages

Stages are where interdisciplinary activities occur: each stage is composed by recommended and cross-functional activities, often done in parallel to increase speed to market, carried out by a team whose members come from different functional areas.

- Stage 0 – Discovery: activities aimed to detect opportunities and formulate new ideas
- Stage 1 – Scoping: the aim of this phase is the evaluation of the project and its market prospects. An analysis of the competition is also made to understand if the project really worth going forward.
- Stage 2 – Build business case: this is the critical phase where it is decided whether the project go on or not. A solid analysis here is mandatory for the company, because it is directly related to the successful continuation of the project. Feasibility studies are conducted on technical, business and marketing sides. This phase is composed of three steps: product and project definition, project justification, and project plan.
- Stage 3 – Development: plans made in Stage 2 are executed and become concrete deliverables. The new product is developed with its design, the plan for manufacturing phases is set out, the marketing launch is planned, and the tests to pass to stage 4 are decided. What is crucial in all these operations is that the team must follow the established goal of the project.
- Stage 4 – Testing and Validation: this phase aims to validate the entire project, including: the product, the production process, customer acceptance, and the financial

aspect of the project. Three types of testing are provided in this phase: *near testing* to detect bugs or issues, *field testing* to obtain feedback by users, and *market testing* to evaluate forecasts of sales.

- Stage 5 – Launch: in this phase the full production and launch of the product on the market occur.

Stages present a similar structure composed of three elements:

Activities: are the work the project team must undertake also in terms of information collection in order to reduce risks and uncertainties.

Integrated analysis: cross-functional team members communicate on the results of all the functional activities and produce an integrated analysis.

Deliverables: results of the integrated analysis are presented for the next gate.

Gates

After each stage there is a gate, where *Go/Kill* decisions are taken to continue the project or stop it. In these points of the process, *gatekeepers* control the quality of the project, decide to make it continue or not, analyze its risk, prioritize tasks, plan together with the team the work to reach into the next stage, and allocate resources.[29] Gates' aim is also to detect eventual failures, in order to avoid any resource waste and consequently to decide whether adjust the project or terminate it allocating those resources to another better project.

The structure of gates is similar for each one:[30]

Deliverables: a review on the work completed in the previous stage by the team is done. What the team should deliver in a deliverable is established at the output of the previous stage, and is based on a standard framework.

Criteria: the project is assessed on the basis of some criteria, which include *must meet criteria* (a checklist to quickly decide if a project can go) and *should meet criteria* (organized in scorecard

[29] Cooper R.G., (2009). "How companies are reinventing their idea-to-launch methodologies", J. Product Innovation Management.
[30] Cooper R.G., (2008). "Perspective: The Stage-Gate Idea-to-Launch Process – Update, What's New and NexGen Systems", J. Product Innovation Management.

to easily prioritize projects), in order to decide efficiently whether the project worth its continuation or not.

Outputs: include a decision on the project (Go/Kill/Hold/Recycle), an accepted program for the following stage, the list of deliverables expected for the following gate, and a date for it.

Gatekeepers are a team of senior management individuals in the business, which is responsible for the go/kill decisions at gates and for an effective allocation of needed resources to projects. They meet at every gate of a project from its beginning to its end, making crucial decisions on going on with project or terminate it on the basis of the information available at the moment.

Gatekeepers use above mentioned criteria to evaluate the work done in the previous stage and can give the team additional business and technical information.

In order to make gates work at best, a good methodology should include guidelines and checklists to avoid both gatekeepers miss critical steps in the decision-making, and project management team wastes too much time preparing gate review reports.

Checklists should ensure an answer to questions on the situation, in terms of time and cost, such as where the team is in a given moment, where it is going to go, what are the risks, whether the management can do something to improve the situation, and so on. It is clear that the role of project manager can never overlap with the role of gatekeeper.

Benefits and disadvantages of Stage-Gate process[31]

Using Stage-Gate process bring to attention some main benefits.

Firstly, project management here becomes a structured job. Secondly, through checklists and guidelines Stage-Gate makes possible a standardized planning, scheduling, and controlling. Thirdly, since the decision-making process is carried out by gatekeepers in compliance with precise rules, it becomes a structured process where there is no place for individual or reckless decisions.

[31] Kerzner H.R., (2013). "Project Management: A Systems Approach to Planning, Scheduling, and Controlling", Wiley.

Despite the improvements brought by Stage-Gate system on traditional project management, it has some pitfalls.

Firstly, even if gatekeepers find out that a project does not deserve continuation, they often hesitate to stop it because they are afraid to lose the resources used until that moment.

Secondly, project team is not given information that can be crucial for they work, unless gatekeepers decide to share it with them. This can affect project team efficient work.

Thirdly, Stage-Gate process makes inevitable that the project team focuses more on producing documentation to give proof of its good work when gates approache than on concentrating on the work during stages.

In the light of the above, it should be recognized that the Stage-Gate is a process just aimed to provide a framework for the project management methodology, and that it is not an end result sufficient in itself. For this reason, this model evolved and changed according to the different needs of companies.

1.4.2 Evolution of Stage-Gate

Since its original formulation in 1990 as a "five stage-five gate" model, Stage-Gate has evolved to a second and a third generation model. Changes resulted from the need to overcome the sequential design and the pitfalls of the original system. What came out was an advanced version of the model that provides for fluent and overlapping phase transitions, removing time wastes at gates between stages, due to long waits for gatekeepers' decision.[32] At each gate, actions proper to gatekeepers, such as go/kill decision, process control, and quality control, can be undertaken simultaneously. The goal of this evolution of Stage-Gate model is to promote speed and flexibility in the product development process.

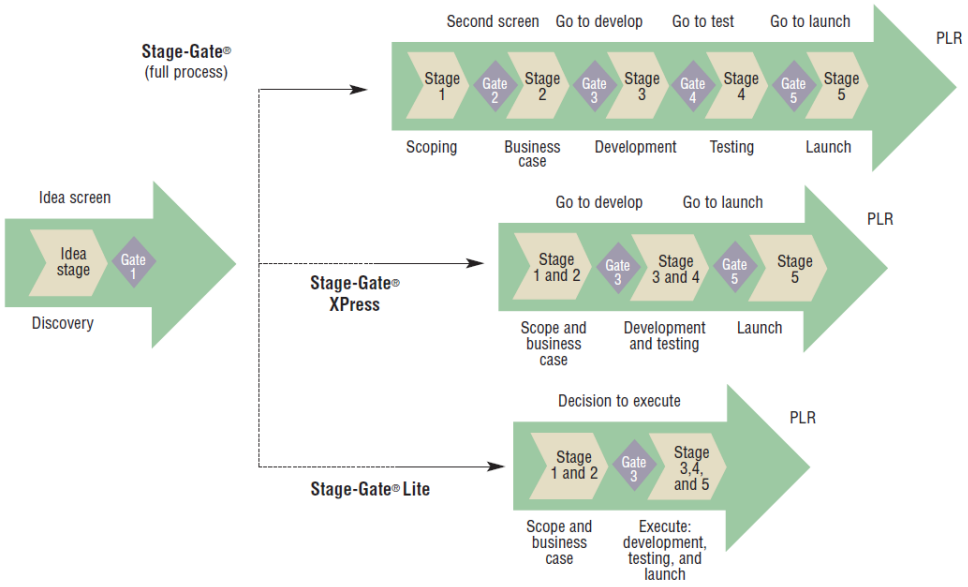
Despite Stage-Gate model provide for a rigid structure in theory, in practice each company adapted the model to its own needs varying the number of stages and gates, depending on the projects they had to manage.

[32] Leithold N., Haase H., Lautenschläger A., (2015). "Stage-Gate for SMEs: a qualitative study in Germany", European Journal of Innovation Management.

The Stage-Gate system became scalable when it began to adapt to this increasing flexibility, offering other different versions of the same model according to project types.[33]

Therefore, besides the already explained Stage-Gate Full, Stage-Gate Lite and Stage-Gate Xpress were created to satisfy the need to manage smaller projects with lower risk, so that they could be developed faster and in a leaner way: these new versions (Figure 8) are called *Next-Generation Stage-Gate*.[34]

Figure 8 - The next-generation Stage-Gate is scalable



Source: Cooper, 2006

Stage-Gate Xpress unifies the first two steps, that are Scope and Business case, and the following two steps as well, that are Development and Testing, leaving isolated only the launch phase.

Stage-Gate Lite, instead, unifies to the phases three and four also the product launch.

Lite version should be used for those product innovation projects focused on a single modification that brings very small changes to the original product and there is a single, while Xpress version can be used for projects that bring visible changes to the original product and have more than one target customer.

[33] Cooper R.G., Edgett S.J., (2005). "Lean, Rapid, and Profitable New Product Development", Product Development Institute.
[34] Cooper R.G., (2006). "The seven principles of the latest Stage-Gate method add up to a streamlined, new-product idea-to-launch process", Product Development Institute.

As seen, in the next generation Stage-Gate, stages and gates became more guidelines than mandatory phases. In fact, according to Cooper, now projects go from milestone to milestone, where a milestones are points of the projects where is confirmed or not the adherence to a set of standards previously established by responsible executives or clients. In this context the role of gates then overlap with the role of milestone, where the go/kill decision occur.

This evolution of Stage-Gate system lays the foundations for introducing another way for managing projects: Agile.

1.5 A better way to manage software development projects: Agile

As seen in the previous paragraph, Stage-Gate is a process ideated for new product development processes. However, despite the relevance of the model, there is a field in particular where project management methodologies for physical product are hardly applicable: it is the software development industry.

Concerning this field, a different project management philosophy came out in response to the rigidities of traditional project management: Agile. More and more companies are turning to Agile development methods because the traditional type of project management is not succeeding. "Agile has evolved through the years from manufacturing processes such as *just-in-time* (JIT). JIT aimed to reduce waste and over-production by determining which parts were needed by the customer at each stage, rather than mass producing too much product that sat in a warehouse"[35].

JIT emerged as a result of the improvement of inventory management, the ability to predict how many items would be required, where and when. Before JIT, it was necessary to hold large stocks because no one knew for sure what quantities would be required, and running out of something was worse than having inventories or surpluses. After JIT, items would arrive at the right moment, responding to the needs.

This concept was enlarged and refined over time, until it became useful also as a tool for software development projects, which need a particular management.

[35] Scanlon Thomas E., (2011). "Breaking the addiction to process - An introduction to agile project management". IT Governance Publishing. P. 16

Agile is similar to *just-in*-time philosophy because software components are tested and delivered as soon as they are complete, rather than being kept on the side and being released when everything is finished.

Agile philosophy and methodologies are going to be analysed in depth in the next chapter.

1.6 Conclusions

For many decades traditional project management has been the only way projects were carried out. The waterfall method imposed the entire project planning in advance, before the development began, and no changes were supposed to be made, causing an overall rigidity of the project life cycle. The change of direction concerning project management happened when the market increased its pressure on companies. It was when they began to face tougher national and international competition and increasingly demanding customers that they realized their need for shorter product cycle-time and consequently a different way to manage projects. Thus, the need for an effective project management to go along with the changing world and gaining a competitive advantage rapidly emerged.

Since the industry was still concentrated mainly on physical product development, the first method that came out was Stage-Gate process, which brought a radical change in the way projects had to be managed. It introduced a well defined process structure where the management was made able, through checklists and guidelines, to implement a standardized planning and scheduling for developing activities, and to control each stage with regulated procedures. As years passed by, Stage-Gate evolved and became scalable and flexible to meet the needs of the different project types, allowing each company to adapt the model to its own projects' requirements. Thanks to its flexibility and capability to improve new product development processes, Stage-Gate is still largely used.

Later, the need for a new project management methods emerged also in the software development industry, where the Stage-Gate did not fit well since it was a methodology ideated for physical product. Thus, a different project management philosophy came out in response to the rigidities of traditional project management concerning this field: Agile.

CHAPTER 2. AGILE

2.1 What is Agile

As seen in the previous chapter of this dissertation, managing a project is an incredibly complex and challenging task to perform. *Agile* was created because of the need of a response to the failure of traditional project management in the software development environment.

Agile Project Management was created to overcome the difficulties and the rigidities coming from 'heavyweight' methods: the stage-based approach implies that since every single phase is implemented in a sequential way, to start a new phase the previous one must be completed. Focusing on software development industry, over the years programmers found out that sometimes there is the need to go back to the previous phase, revise and modify something, and then going on with the new one. It provides for testing phases where it is possible to make corrections to plans.

Specifically, agile software development defines some principles helping cross-functional teams in organizing and collaborate in order to find solutions to develop a software in compliance with requirements.[36] It encourages adaptation and evolution on both planning and development activities, frequent deliveries, and high flexibility to changes, which may occur during the project. Agile principles gave life to many different software development methods, which are in continuous evolution.[37]

Agile is successful since it understands that software development involves people dealing with difficult and continuous problems which need to be solved, and is useful since stakeholders' feedback on the product is used to continuously improve plans. In most cases, feedback is not only highly effective but are necessary in the long term as a continuous confirmation by the customer on the coherence of the software to his expectations. [38]

[36] Collier K., (2012). "Agile Analytics: A Value-driven Approach to Business Intelligence and Data Warehousing", Addison-Wesley.
[37] Larman C., (2004). "Agile and Iterative Development: A Manager's Guide", Addison-Wesley.
[38] Carroll J., (2012). "Agile Project Management for speedy results", EasySteps Limited.

2.2 The Agile Manifesto

The Agile Software Development Manifesto was created in February 2001, when 17 people -17 methodologists - recognized the difficulty of creating good software and wanted to instill new values into software development teams. They had a meeting in Utah's mountains where they spent few days together. Those people had one goal in common: to find an alternative to the existing software development processes which resulted heavy and static. This group of people, who gave itself the name 'the Agile Alliance', was composed of software professionals with an independent thinking who were able to create, with different and sometimes opposite ideas, the Manifesto that we know today. [39]

The Agile Manifesto is the following:

"We are uncovering better ways of developing software by doing it and helping others do it. We 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." [40]

First of all, it is useful to analyze in detail every single aspect of the Manifesto before going further with the discussion. First of all, the four bullet points have an order and a deep meaning: in each bullet point the first part indicates a preference according to the Agile method and is emphasized, while the second part of the sentence describes an item that, even if important, is of lesser priority. The first part emerges compared to the other one. This distinction is not random but it lies at the heart of agility, since Agile method asks people to list what is more valuable for them and what is essential for the project to come out and succeed. To summarize, the Manifesto defines values, not alternatives, encouraging developers to focus on certain areas but not eliminating others.

The first concept that comes out from the Agile Manifesto is the word ***uncovering***: this group of 17 people, experienced and well known in the software development sector, is trying to give

[39] Carroll J., (2012). "Agile Project Management for speedy results", EasySteps Limited.
[40] Fowler M., Highsmith J., (2001). "The Agile Manifesto", Software Development, 9(8), 28-35.

a framework and some guidelines but at the same time they point out that they do not have all the answers.

Secondly, a highlight is deserved by the expression ***by doing***, which is used to indicate that the Alliance members actually practice these methods in their own work.

Thirdly, the group is about ***helping***, not just telling. The Alliance members want to help others through agile methods, and later improve their knowledge by learning from those they tried to help.

An in-depth analysis of the meaning of the four statements is given below.[41]

Individual and interactions over processes and tools

In this statement, the Alliance recognizes the importance of processes and tools, as the Manifesto is not intended to eliminate areas, but it states that the interaction of skilled individuals is of even greater importance. What is more important in this point is teamwork and communication, the fact that people work together and interact between each other is emphasized and is the heart of software development systems. The Manifesto does not say that tools are not important, they are indeed still key in the development software, but the tools should be combined and should fit the team's needs. This statement answers to the question 'Who do you think will develop better software? Five software developers with their own separate but unique tools working together in a single room, or five isolated individuals with a well-defined process, a common set of the most sophisticated tools and a huge office?' as Tim Hinds pointed out in his article "The four values of Agile Manifesto" [42]. The answer is that the team will perform better and this is the meaning of this first bullet point.

Working software over comprehensive documentation[43]

In the traditional way of carrying out projects, a lot of time is spent documenting the system. The team spends months documenting requirements, analysis, design, and test cases, leaving less time for the development of the product. This is one of the biggest obstacles to progress

[41] Awad M.A., (2005). "A comparison between Agile and Traditional Software Development Technologies", University of Western Australia.
[42] Hinds T., (2014). "The four values of Agile Manifesto", Neotys Blog.
[43] Williams C., (2013). "Agile Manifesto: working software over comprehensive documentation", Source Allies Blog.

in a software project since there are some drawbacks. In today's business economy, where rapid changes are the order of the day, spending too much time on producing documentation can cause that when all the documentation is ready to start the development, it will be out of date and need to be started again.

Another drawback is that teams spend a lot of time in designing parts of the system and producing respective documentation before the project begins, just to discover in the end that these features were not actually needed. In other words, in this way teams waste effort and time by documenting features that may never get used. Here too, the Manifesto does not say that documentation is not needed, indeed it is, but it should not hinder the team's progress and at the same time it should provide value. It is suggested that the documentation is composed by automated tests which are updated as the system evolves like unit tests (individual routines), functional tests (routines in the system), and integration tests (whole system).

To sum up, comprehensive documentation is not negative per se, but the primary focus must remain on the final product, that is delivering working software. Documentation is a great resource for users or co-workers who are wondering what the software is and how it works. If the company is truly Agile, it needs to remember this value: the primary goal of software development is to create software rather than documents.

Customer collaboration over contract negotiation.[44]

Agile pioneers emphasize the supplier and customer confrontation and collaboration, rather than the mere negotiation of contractual terms. It means that the customer is part of the project; customer and developers create a partnership in which they together discover, learn, and make some changes to plans according to opinions and preferences. In this way, the customer can check the project step-by-step, rather than only when it is finished. This kind of collaboration may seem difficult to establish and maintain, but it leads to substantially better results. Contracts and project charters are necessary as they provide conditions and clauses within which the parties should work, but only through ongoing collaboration the team can be able to understand, produce, and deliver what the client expects. It is easy to understand that

[44] Ambler S.W., Holitza M., (2012). “Agile for Dummies”, John Wiley and Sons.

the most important element is the customer's willingness to actively participate, since he is the one that can tell the team what are his expectations from the project. At the same time, the development team should make its best in understanding what the customer communicates and adapt its work to his requirements. Successful development teams work closely with their customers and build a frequent communication throughout the entire project life cycle, from the negotiation phase until the end. This is done in order to avoid changes after the project's completion, since they are difficult to implement without compromising the entire work done. The customer does not have the expertise to tell the development team how to carry out the project, but it is crucial to collaborate with him and get his feedback in order to develop a software in compliance with his requirements.

Responding to change over following a plan[45]

Because of its rigidity, traditional project management is not able to make the project respond to changes during the development phase, and neither it can accommodate new product requirements. Therefore, following a plan is a positive practice because it allows the team to be organized, especially at the beginning of the software development when everything is yet to be determined, but often following a plan means being 'stuck in it' and can make the project miss opportunities to be better.

The world of business and technology is turbulent, because of changes in technology, business trends, customer's needs. New ideas or problems due to these changes arise during the project development. In this scenario, the Agile project management approach allows to incorporate such not planned changes into the ongoing work, and use them to bring additional value to the project. A successful plan needs to be malleable and responsive to changes that might happen during the software development. The ideal plan should allow room to quick changes and the flexibility of Agile approach helps in increasing the project stability.

In this point of the Agile Manifesto, the reference to the first value is clear; being responsive to changes needs an effective communication between team members and customer. Here much responsibility is in charge to testers, those who find bugs or missing parts of the software. Their findings can lead to start a discussion with the customer about changes or adjustments that

[45] Apke L., (2015). "Understanding the Agile Manifesto. A brief & bold guide to Agile", Lulu Publishing.

should be made in order to increase the software quality and usability. A proactive approach is crucial when testing, because the team should be prone to actively go along with continuous changes and improvements.

2.3 Agile principles

The Agile Manifesto, also called 'Manifesto for Agile Software Development', besides proclaiming the four key values analyzed above, it is a formal proclamation of 12 principles which guides an iterative and people-centric approach to software development.

Agile software development focuses on keeping code simple, testing continuously, and delivering functional bits of the application as soon as they are ready.[46]

The 12 agile principles are a set of guidelines for supporting project teams in the implementation of agile projects. These principles have been adapted for managing a variety of business and IT-related projects. They are the following[47]:

1. *Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.*
2. *Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.*
3. *Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.*
4. *Business people and developers must work together daily throughout the project.*
5. *Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.*
6. *The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.*
7. *Working software is the primary measure of progress.*

[46] Martin R.C., Martin M., (2006). "Agile Principles, Patterns and Practices in C#", Prentice Hall.
[47] Fowler M., Highsmith J., (2001). "The Agile Manifesto", Software Development, 9(8), 28-35.

8. *Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.*
9. *Continuous attention to technical excellence and good design enhances agility.*
10. *Simplicity--the art of maximizing the amount of work not done--is essential.*
11. *The best architectures, requirements, and designs emerge from self-organizing teams.*
12. *At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.*

Each one deserves a more detailed explanation of its meaning, which is given below.

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

The first agile principle underlines the importance of developing a 'valuable software'. The matter here is to understand what is a valuable software, since it is a subjective concept. For example, from the point of view of a developer, a valuable software should be nice designed following all the principles of encapsulation. From the point of view of the end-user, a software to be valuable should be rich and customizable according to his preferences and it should have many features that make the daily work easier. Another perspective it that of the support department, who has other priorities: for the team a valuable software is that one easy to install, configure, manage, and cost effective in maintenance. It is clear that there are many different and sometimes clashing interests, which need to be balanced. Therefore, a valuable software is able to support the goals and maximize its added value, but mainly it must satisfy the final customer, because only a long list of satisfied customers can help software development firms to survive.

In this principle, also the word 'continuously' deserves an highlight; it means that the client not only will be given the final project, but he will also be involved throughout its implementation, giving feedback, comments, and preferences. Developers realized that it is not possible for a team to develop a software in isolation and then deliver a finished product that is compliant with client's requirements. In order to achieve this result, it is crucial to build a continuous relationship between the team and the client, in which each one brings ideas aimed to refine the features and deliver an improved, effective software. Outside the Agile approach,

developers use to deliver the software to the client at its completion, in the end of the project. This practice reveals to be costly, since changes made on a completed software are more difficult than changes made during the development phase.

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

One of the most important points of Agile is that changes are inevitable; requirements will change during the software development project, therefore developers should welcome and embrace them. This principle states that changes are accepted even when they come late, since software must be responsive to changes. In this way, it might be more difficult to leverage on its added value, but an evolving software will surely enhance the final result. As said above, it is clear that it is better to change requirements before relative features are implemented, when all the parts are still planning the work. Anyway, in nowadays market where being competitive is the key to success, it is crucial to be responsive to changes, at any time they emerge. Furthermore, it has to be considered the volatility of technology; while a project is executed, new technological development can bring mandatory changes in the software development sector.

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

The third principle of the Manifesto states one of the main goals of Agile approach: the frequent delivery of working software at regular intervals. To reach the goal, the preference should be on the shorter time schedule, in order to make the work iterative and able to improve step by step. In Agile, iterations should be long enough to enable the team to deliver a tangible incremental part of working software, while short enough to allow frequent feedback which makes sure that the project is compliant with client’s requirements.

In traditional, linear approach, projects run the risk of not being synchronized with the reality, because by the time the project is completed two main drawbacks can happen: stakeholders feel less involved in the project, and technological progress has not been kept into account. Through frequent delivery of working parts, stakeholders are asked for feedback; in this way they will feel more involved and cooperation will increase. Another critical factor to consider is

the environment in which the project is carried out, for example a change in the organization's structure can happen (e.g. mergers or takeovers). This can lead to a need for changes in business strategy, competitive circumstances, new regulations, and laws.

To summarize, the short-time schedule is helpful for both the client and the development team; frequent feedback and reviews can be used by the team to elaborate and propose prototypes and ideas to stakeholders, in order to make clear what are the required features to implement, and avoid consuming resources. In addition, schedules make the project status more transparent for all the stakeholders, while they have the opportunity to provide improved direction to the development team.

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

The fourth principle of the Manifesto repeats the importance of the communication between stakeholders, a concept already largely seen above. This sentence specify that collaboration should be on a daily basis, in this way a good synchronization can be established between team and customer's needs. Through this kind of communication all parties can feel more confident and closer to each other. Business people and developers who do not so often communicate, will have a huge disadvantage in comparison with a team were all the parties are highly involved.

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

The first key concept to highlight in this statement is the 'motivation' of developers, which is to be intended in two ways: motivation concerning their expertise, and motivation related to the implementation of agile methods. It is important that the project manager focuses on team's skills, cooperation, and motivation, giving them the possibility to experience and train, since software development is a specialized form of knowledge.

Another important concept to focus on is 'trust'; developers should be trusted since they are supposed to have a deep expertise and all the capabilities of getting the work done. Therefore, a manager should motivate the team members and give them the adequate tools, offering a good and pleasant working environment able to support the needs of the entire team. The

team should be given the responsibility to deliver the project and the authority to do what is necessary, in order to be more connected to the whole project and care more about its success.

To sum up, this statement stresses that it is important to trust team members, let them take the right decisions, and make them aware of their responsibility. A reward system can be provided for individuals who made extra efforts or solved complex problems. Motivation is one of the key success factors for agile projects.

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

The importance of communication in Agile has already been largely stressed, but this statement adds that it should be face-to-face. The maximization of interpersonal communication will bring some advantages for the project development. First of all, time and effort required for this form of communication is lesser than others, because going to the direct source of the information is easier, more effective, and less time-consuming than, for example, e-mail communication.

Face-to-face conversations are more detailed because body language, eye contact, and voice tone play a crucial role when building trust, confidence, and shared understanding. Face-to-face makes the team more united, and team members become more open to other individuals' ideas.

7. Working software is the primary measure of progress.

The seventh principle of the Manifesto states that the progress can be measured only on the basis of completed tasks. Working software means that it has been successfully tested, delivered, and accepted by stakeholders. [48]

In general, in the world of large firms, people tend to measure and track everything that can give estimates. Anyway, even if it is a good practice, measuring too much leads to waste time and miss the focus and the aim of the job, which is to develop a product. Numbers cannot track how much value the team is actually bringing, and whether or not it is solving customer problems through their work.

[48] Highsmith J.A., (2002). "Agile software development ecosystems", Addison-Wesley Professional.

To sum up, this principle states that the final goal should be the delivery of a working software. Velocity, numbers, deadlines, and story points are not the measure of progress; the only thing that matters is how much working software has been created at the end of the iteration. [49]

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

The expression 'sustainable development' means that the software should be created keeping team's concentration at its best. It is done by carrying out the work carefully and steadily, maintaining a constant pace, as written in the second part of the statement.[50] The team should invest a constant amount of time and effort throughout the project realization, and this applies not only to developers, but also to all stakeholders, end users, testers, and designers. If this way of proceeding is ignored, especially in projects with long deadlines, it can happen that people start to work relaxed at the beginning, and when the deadline gets closer they start working nights and weekends.

To sum up, in order to promote a sustainable development, the whole team, stakeholders included, should work together from the very beginning of the project and give their best for its realization. The project manager should favor team's communication and discussion and avoid any undesirable attitude. Everyone involved in the project should follow the same productivity pace which foresees a balance between free time and relaxation, and focus on the project.

9. Continuous attention to technical excellence and good design enhances agility.

What is highlighted in this ninth principle is the technical excellence, which should be improved day by day; this is the reason why the team is asked to stay focused on the daily business. This statement is slightly connected with the previous principle, which said that the team should keep a constant pace and stay focused from the very beginning of the project realization. Agile

[49] Van Amerongen R., (2008). "Agile software development, the principles. Principle 7 [...]", Amis Technology Blog.

[50] Van Amerongen R., (2008). "Agile software development, the principles. Principle 8 [...]", Amis Technology Blog.

teams usually spend all the necessary time and effort to understand how to solve problems, how to do things at their best, and how to deliver shippable solution at every step. [51]

Many people think that agile software teams just put together codes without a design in mind, but this is false. Agile understands that work should be done properly in order to avoid late, and consequently difficult, modifications and adjustments. Technical excellence means to gather the right individuals with the right skills to create the best product possible, so Agile recognizes the importance of quality. At the same time, good design is important and should be the starting point of all projects.

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

This principle tells agile teams not to build more than the minimum required and not to go further that the step. Developing a software step by step implies doing it in a simpler way. In a sentence, this principle encourage developers to focus on today's job, avoiding to think too early about features that are steps away; it will progress day per day.

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

In the past division of labor, the roles, the sum of different parts done by designers, architects, and developers, were put together at the end of the project without any interaction; in this way too often things did not work. Instead, agile projects are focused on self-organizing teams: it means that estimation and planning should be at least on team level. What is crucial here is that the team members work together, learn from past mistakes, prevent problems that might occur, and face new ideas and challenges as a whole system. In this scenario, it is clear that communication makes sure that the team is organized correctly, and trust and confidence are crucial values. The team does not need a single person in charge of the project and responsible for its success: teams in which all the necessary skilled members are present, are also able to organize themselves without one person on top.[52]

[51] Apke L., (2015). "Understanding the Agile Manifesto", Lulu Publishing.
[52] Fowler M., Highsmith J., (2001). "The Agile Manifesto", Software Development, 9(8).

12. At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

As seen above, agile methods are flexible as they foresee the possibility for developers to modify the project according to customer's needs, occurred problems, or environmental changes. This last principle states that it is important for teams to review the done work and the work that is yet to come, at regular intervals.

This principle underlines the importance of agile retrospective; this activity should be done at the end of each iteration, that coincides with the beginning of the following iteration, and whenever the team requires it to be done. This process helps teams to improve the product while they are working on it. Retrospective helps in checking if the planned tasks are completed, and reflects what events occurred since the last meeting. Retrospective should be done at regular intervals in order to satisfy some requirements of Agile approach: frequent feedback, communication, collaboration, progress measurement, technical excellence pursuit, organization.

2.4 Agile methods

The developers who drafted the Agile Manifesto and its Principles gave life to some development methods based on this approach. Some of the main methods are going to be analyzed in the next section, such as Extreme Programming, Crystal, Dynamic Systems Development Method, and Kanban.

Later, it will be explained that Ken Schwaber and Jeff Sutherland originated the Scrum method, the most widely used agile method for software development, and not only. For this reason, an in-depth analysis of Scrum is going be made.

In the previous section of this dissertation, the focus was on the Agile Manifesto and Principles, which represent the framework on which agile methods have been created.[53] Now the focus will be on the different methods, their advantages, weaknesses and how they differ from each other.

[53] Stellman A., Greene J., (2015). "Learning Agile: understanding Scrum, XP, Lean and Kanban", O'Reilly Media.

2.4.1 Extreme Programming method (XP)

Created by Kent Beck, Extreme Programming is the oldest agile methodology, defined by its author as a software development methodology aimed to enhance the quality of software and the responsiveness to changing customer requirements.[54]

Extreme Programming, in compliance to agile principles, is an iterative and incremental process, which provides for a division of the project in smaller releases representing each one an increase in product functionality. Releases are planned almost frequently, usually from one to three months, and are discussed and planned in the so called 'planning games' by both customers and developers. Releases contain iterations, each one lasting usually for three weeks, used to facilitate planning deadlines.

This method is very simple and concrete and it tends to emphasize teamwork: managers, customers, and developers. It provides to the developers profound and detailed discipline on how the work should be done.

Extreme Programming is based on four control variables[55] that should be considered in software development. The first variable is **cost,** which represents the amount of resources, not only money, needed for the project. The second one is **time**, intended as the deadline for the system delivery. **Quality** is the third variable, and it is related to the degree of compliance of the product with customer's requests. The last variable is the functionality or **scope**, which is related to the control variables but also to the 'values'[56], mentioned below. Each variable is closely related to the others, since, for example, an increased requested quality or a reduced amount of available resources may bring an increased time needed to complete the project.

Extreme Programming is based on four underlying core **values**, that are intended as guidelines for developers.

The first value is communication between developers and customers, which is one of the key aspects of this methodology, and it includes direct communication or written documents in order not to forget important issues about the project.

[54] Beck K., (2005). "Extreme Programming Explained: Embrace Change", Addison-Wesley.
[55] Dudziak T., (2000). "eXtreme Programming, an overview", Methoden und Werkzeuge der Softwareproduktion WS.
[56] Mnkandla E., Dwolatzky B., (2004). "A survey of Agile Development Methodologies", the transactions of the SA institute of electrical engineers: 236 - 247.

The second value is simplicity, in both systems and methodology, intended as the search for the simplest and lighter way to carry out the work to project completion. Extreme Programming developers tend to simplify the development by avoiding useless artifacts, in terms of user stories, plans, and code, which are not needed for demonstrating some aspects to the customer. It can be said also that communication and simplicity could be seen together because the simpler the system is, the easier will be communication.

The third value is composed by feedback from mangers, stakeholders, and developers, which helps to ensure continuous quality and decreased development time: if feedback is continuous, problems and changes that can occur will be smaller, cheaper, and easier to solve.

The fourth value is courage, as the name 'extreme' suggests, intended as the aggressiveness needed, from both developers and stakeholders, to implement agile rules and practices against more traditional and wiser methods for software development.

These are the value at the basis for all the practices of Extreme Programming development, designed to allow changes at any stage of the software development process. The word that summarize XP is 'simplicity', and it is a method generally preferable on small and highly dynamic projects, even though it is open to customization for bigger projects: its simplicity make XP be easily scalable.

2.4.2 Crystal method

Crystal methods are a family of 'lightweight' agile methodologies developed by Alistair Cockburn in 1998. The name Crystal has been chosen because the faces of the gemstone recall the many different point of view possible on the underlying essence of principles and values, in fact faces represents tools, techniques, and roles.

Crystal methods are based on the concept that each project is unique and dynamic, therefore each one require a different methodology providing for tailored policies, practices and processes in order to meet the project's needs. Thus, Crystal provides for adaptable approaches to software development according to their size and complexity, represented by the colors of

geological crystal's hardness: Clear, Yellow, Orange, Orange Web, Red, Maroon. The larger and critical a project is, the darker is the color of methodology fitting to it. [57]

The above-mentioned key concepts, namely size and complexity deserve a further explanation: size is the number of people involved in a project, thus if the team increase Crystal should change and provide a more formal management structure and documentations. Complexity, or criticality, is one of the aspects considered uniquely by Cristal methodologies, and recalled by the 'safety' principle mentioned later: Crystal recognize that the risk of damage to end users is different in each project, for example a malfunctioning on a medical life support system is much more dangerous than a video game. Criticality also helps in identifying the scope of the project, where the team should concentrate its work.

Each Crystal methodology is based on seven core properties, independently from the color[58]: frequent delivery, continual feedback, close communication, personal safety, focus, access to expert users, and automated tests and integration. Cockburn saw that the more of these key principles are present in a project, the more probable it will be successful.[59]

According to Cockburn, Crystal methods respond to issues about the accuracy of the model and its capability to meet customer's needs. Indeed, Crystal methods focus on communication, claiming the primacy of human interaction especially between team members, since information flow is the key to success. A paramount importance is recognized also to involvement of users, community and people. Projects developed with Crystal methods follow an iterative development process, as it happens for other agile methodologies.

Crystal provides for many methodologies, each one with different characteristics according to project size and criticality. As one of the two factors grows, Crystal provides for mechanisms able to provide better support to increasing burden.

[57] Cockburn A., (2007). "Agile Software Development: Software development as a Co-operative game", Addison Wesley.
[58] Coffin R., Lane D., (2006). "A Practical Guide to Seven Agile Methodologies", Part 1, XP, Scrum, Lean, and FDD.
[59] Highsmith J., (2002). "Agile Software Development Ecosystems", Addison-Wesley Professional.

2.4.3 Dynamic Systems Development Method (DSDM)

Dynamic System Development Method was developed during the 90s, and is a methodology characterized by fixed time and resources, and establish functionalities to develop accordingly. DSDM is based on five phases:[60]

1. Feasibility Study: where it simply has to be decided whether to use this method or not, considering the type of the project and potential issues.
2. Business Study: where it is made sure that everyone understand the business scope of the project, and the first early plans are made.
3. Functional Model Iteration: where the activities of analysis, prototyping, and coding begin and are reiterated in order to obtain a prototype code and analysis models.
4. Design and Build Iteration: where the software is actually developed. In this phase users review design and prototypes and, in case, help in adjusting future development.
5. Implementation: where the software is delivered to users.

The model is iterative, thus it requires the completion of a stage before going on with the following one.

DSDM requires the adherence to nine key principles,[61] which seem to be essential for a correct implementation of this method.

- Active user involvement: is the most important principle since it helps reduce potential errors due to misunderstanding on end user's needs.
- Empowered and self-managed teams: so that authorizations are not needed and changes can be done faster.
- Frequent delivery: helps to ensure that errors are detected quickly.
- Delivery of required functionalities: it is imperative that the delivered software meets customer's needs to be accepted.
- Integrated testing: during the entire lifecycle phase to maximize testing opportunities

[60] Awad M.A., (2005). "A Comparison between Agile and Traditional Software Development Methodologies", University of Western Australia.
[61] Voigt B., (2004). "Dynamic System Development Method", University of Zurich.

- Collaborative approach between technical and business members: so that trust and cooperation increase.
- Iterative and incremental development: the job is implemented through small steps which increase the ability to respond to changes and to add new features
- Reversibility of changes: according to changed prioritization.
- Establishment of high-level requirements: there are requirements that can never change and should remain fixed even if stakeholders ask for their variation.

As previously said, DSDM method is characterized by fixed time and budget, thus each project must be split into parts each one with a previously fixed number of features, and precise time limit and budget. If the project is near to run out of resources, intended as time and money, the less important features are abandoned or put aside. In this scenario, it is clear that a correct prioritization of features is paramount: DSDM employs a strict method for prioritizing requirements, following the so called MoSCoW rule which categorize features as *Must have*, *Should have* (if possible), *Could have* (if it does not impact more important ones), and *Want to have* (if resources allow it). User feedback as well is crucial, that is why this method assign prototypes a paramount role since early in the project phase to gather feedbacks.

Since DSDM is a really business-centric practice, it is not to be used for projects representing a risk for users' safety as it focuses on business value and not on risk at all.

2.4.4 Kanban Method

Kanban method, formulated by David J. Anderson in 2005 as a "continuous, incremental, evolutionary process"[62], is inspired by the Toyota Production System developed by Taiichi Ohno during the 40s to monitor steps in the manufacturing process. The name comes from the Japanese word "visual signal", in fact the method is based on positioning particular cards (*kanbans*) indicating small product features on a board divided into columns representing each one a phase of the product development process, the *Kanban board* (Figure 9).

[62] Rossberg J., (2014). "Beginning Application Lifecycle Management", Apress. P.57.

Figure 9 - The Kanban Board with a six steps workflow

Source: Leankit, 2015.

Kanban method is based 3 fundamental principles and 5 properties[63]:

The first principle is that Kanban does not provide for any specific steps or roles, it just tells the team to start from what they know and build the system on those basis, hence it allows to continue using investments done and improve them.

The second principle stands in Kanban's push to continuous, incremental and evolutionary change: the organization should recognize and agree that it is the only way to achieve improvements.

The third principle asks the organization to respect roles, responsibilities and titles in order to eliminate fear and facilitate future changes.

Five properties were identified by Anderson to be the key for a successful implementation of Kanban method.

1. Visualize the workflow and understand how it works will help to improve it and make the right adjustments. Kanban boards are the symbols of this methodology, on these

[63] Wagener J., Schmit S., Mandal A., Rajendran V., (2012). "Project Management Using Kanban", University of Luxemburg.

boards teams put cards representing tasks and work items of the project to visualize a project's current state. Cards can be moved through the board according to their development stage: pending, analysis, development, test, done, and others. Figure 1 shows an example of a six stages Kanban board. This approach helps all the people involved to feel more comfortable and to work in a more confident way, understanding the limit of what can be done.

2. Limit the Work In Progress (WIP): in order not to overwhelm teams and departments, workload in progress should be coherent with team capacity in order to let people focus their efforts on few tasks, and only when those tasks are completed starting something new. This approach helps teams to increase productivity avoiding confusion. Thus, Work in Progress, and then work items, should be limited for each one of the phases of the process workflow.

3. Measure and manage flow during the development. When the workflow is visualized on the board, a crucial task is monitoring it and evaluating effects that changes have on the system. The team should focus on the flow itself and on its interruption: when a stage, that is a column, is full of work items while another column on the right is empty, it is easy to identify potential bottlenecks. A deeper analysis can get data which reveal the problems behind them, and help to prevent any other problem that might occur in the future.

4. Make Process Policies Explicit so that more rational and objective discussions on process improvements will be possible. If everyone understands how things work and how the work is done, more informed decision will be taken. With the process of workflow visualization everyone is really aware of what is being done and what are the goals.

5. Improve Collaboratively (using models and the scientific method): as discussed, the Kanban method encourages small, continuous, incremental and evolutionary changes. If these changes are taken in small steps, it will be easier to overcome any arising problem. In few words, this principle states that if team members are all well prepared about workflow, processes, and risks they will be more able to build a common understanding and a collaboration that help in successfully facing future problems.

This method is seen as a process that never ends, because it requires constant monitoring and analysis: it is the milestone method for a company culture aiming to continuous improvement. As seen, Kanban does not provide for prescribed roles, while other agile methods do, and development is structured on continuous delivery, rather than somehow time boxed. For these reasons, Kanban has the advantage to be very flexible and light, as well as easily visualized thanks to the boards, therefore it is more suggested to manage projects where the team operates is an environment characterized by a high degree of priority variability.

2.5 Scrum

The focus will be now on the agile Scrum[64] as it is the most well-known and widely adopted agile method. The word *Scrum* was introduced by Takeuchi and Nonaka in 1986 as a concept related to new product development, defining it as a form of "organizational knowledge creation, [...] especially good at bringing about innovation continuously, incrementally and spirally"[65]. The name 'Scrum' recall the formation of players in rugby football, in fact this method is designed to involve a cross-functional team working on many overlapping phases of the developing project, where the team work as a unit, passing the ball back and forth, to gain the game. [66]

In 1993, Ken Schwaber and Jeff Sutherland introduced agile Scrum as the method created by Teckeuchi and Nonaka, but applied to software development projects. Sutherland and Schwaber created agile Scrum because they needed a faster, more efficient, and more effective way to manage software creation, a method able to adapt and evolve, in order to overcome difficulties represented by the rigid, prescriptive, hierarchical methodologies used in the past.[67]

Therefore, Scrum was developed as a new and improved way to manage projects, designed to be evolutionary and adaptive to changes that might occur during the development of a

[64] Maximini D., (2015). "The Scrum Culture. Introducing Agile methods in Organizations", Springer International Publishing.
[65] Nonaka I., Tackeuchi H., (1995). "The Knowledge-Creating Company: How Japanese Create the Dynamics of Innovation", Oxford University Press. P. 3.
[66] Takeuchi H., Nonaka I., (1986). "New Product Development Game", Harvard Business Review.
[67] Sutherland J., (2014). "The Art of doing twice the work in half the time", Crown Business.

project[68], and it obtained such a success that Scrum is now the leading agile development methodology, used by the top companies around the world.

This method is very simple, as it offers a framework and practices that keep everything visible in order to allow Scrum's practitioners to know what is going on in every step, and is designed to be flexible during the entire project life cycle in order to allow the team to make adjustments at any time if needed. [69]

Scrum is composed of small steps called *sprints*. Each sprint in general lasts from 1 to 4 weeks: it is long enough to give team members the time to understand what has to be done (design, development, testing) and realize it, but at the same time it is short enough to avoid the technology becoming obsolete and the project not being competitive any more.

Another main element of Scrum is the *Backlog*. It is a list of all the features that needs to be developed in order to complete the project in compliance with requirements. *Product Backlog* helps the team to organize the work starting from the top prioritized items to the lowest ones, in order to develop the most valuable functionality first. Sprint Backlog is a section of the product backlog which lists the features to develop in a specific sprint.

Scrum can be seen as a framework for managing a process, this method does not provide detailed descriptions of how everything has to be done: much is left to the development team because the Scrum team has all the capabilities needed to solve the problems that might arise.

What is different in the Scrum is the fact that there is no team leader taking decisions: the team, all together and self-organizing, operates as a whole. The team is also cross-functional because each team member has different capabilities and is specialized in a specific technical field, and everyone is needed to complete the project. One thing that is important to underline is team size: the recommended size of a scrum team is usually less than 10 people because small teams tend to be more efficient than big teams. The reason lies in the fact that in this way communication is easier, team members can easily interact between each other and with the client. When the team is large, difficulties of communication may arise.

[68] Schwaber K., Sutherland J., (2016). "The Scrum Guide. The definitive Guide to Scrum: the rules of the game", Scrum Guides.

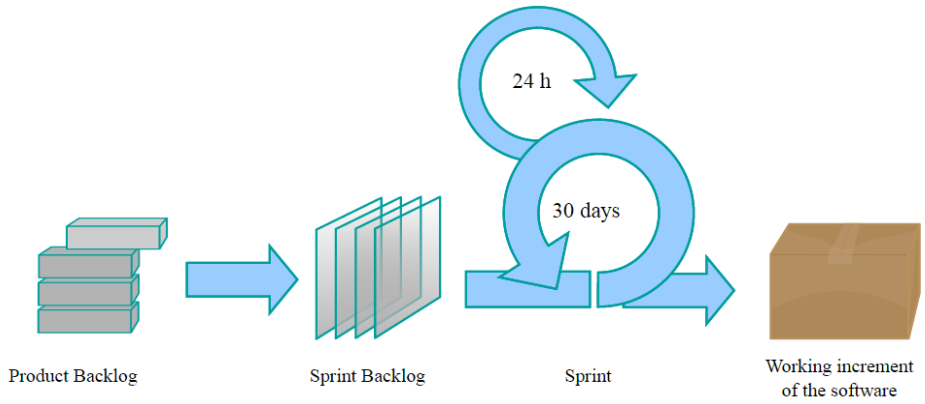
[69] Dönmez D., Grote G., Brusoni S., (2016). "Routine interdependencies as a source of stability and flexibility. A study of agile software development teams", Information and Organization 26(3), 63-83.

2.5.1 Scrum process

The Scrum project is progressed through a series of sprints, which are iterations with a determined time length. Scrum can be seen as an "iterative, incremental process skeleton"[70] (Figure 10).

At the start of each iteration, the team plans the work activities to undertake according to the sprint backlog, which lists the features that should become an increment of functionality by the end of the iteration. The sprint backlog is identified according to the prioritized product backlog. The team considers the assets, resources, and technologies needed to undertake the work, and can arrange for adjustments. Throughout the sprint, the team carries out all the activities aimed to develop the planned functionalities. At the end of the iteration, the team delivers the increment of functionality developed, in order to allow the stakeholders to inspect the work done. They can check and ask developers for timely adjustments or adaptations to the project.

Figure 10 – The Scrum process



Source: Heller, 2011

Usually Scrum keeps into consideration many variables, such as: user requests, in order to improve the current system, time pressure, which is fundamental to be competitive, and competition and quality, in order to see what changes are needed and what resources are available.

[70] Schwaber K., (2004). "Agile Project Management with Scrum", Microsoft Press. P. 5

As said, the organization of the work is divided into sprints, during each one there are four meetings[71] aimed to help team members, Scrum Master, and Product Owner to check the project realization and bring eventual changes.

The meetings are held during different phases of every single sprint in this sequence: Sprint Planning Meeting, Daily Scrum Meeting, Sprint Review Meeting, and Sprint Retrospective Meeting, which is helpful to revise what has been done.

At the start of the sprint, the Scrum team, the Product Owner, and the Scrum Master meet up all together in the Sprint Planning Meeting, where Sprint Backlog is negotiated according to the prioritized Product Backlog. During this meeting, the product owner communicates which items are the most important for the business, deciding also Sprint Backlog prioritization. At the same time, the team calculates the amount of work that can be reasonably done in that sprint, according to the available resources. It is not an easy task because developing teams face complex work and uncertainty, nevertheless they have to be as precise as possible and plan in detail their 'production capacity'. The work that needs to be planned (and hopefully done) in a sprint should be that one able to produce a potentially deployable product increment.

When the Sprint has been carefully planned and the actual sprint execution begins, it is the moment of Daily Scrum Meetings. During one of these meetings, which should not last more than 15 minutes, each team member reports to the others and to the Scrum Master his work of the previous day, what obstacles he faced, and eventually what he needs to overcome such difficulties. If needed, in this phase Product Backlog can be refined during the Backlog Refinement Meetings aimed to help the product owner to reprioritize items.

After the execution phase, it is the turn of the Sprint Review Meeting, in which the team shows what has been done during the just finished sprint and what is the additional functionality created. At this point, the product owner compares the tasks planned in the first meeting with the results, to check what has actually been done. If any incomplete tasks remained, they should be ranked according to their priority and left for the next sprint. Sprint Review Meeting is normally attended also by external stakeholders and is very useful, since as the product grows new requirements or adjustments might arise: iterative development allows the creation of

[71] Rad N.K., Turley F., (2013). "The Scrum Master training manual", Management Plaza.

products with requirements that could not have been specified from the very beginning, like in a plan-driven approach.

After the Sprint has been executed, it can be held a Sprint Retrospective Meeting, in which the team focuses on its own development process. This phase is delicate, since it tends to underline the team difficulties and weaknesses, and it might create a hostile working environment and problems between the members. In order to avoid this, the Scrum Master should offer the team a series of techniques to facilitate the meeting procedure.

2.5.2 Scrum roles: Product Owner, Scrum Master, and Scrum Development Team

A Scrum team is composed by three specific roles: the Product Owner, the Scrum Master, and the development team[72] composed by designers, testers, developers, and other technicians. All management responsibilities in a project are divided among these three roles. Other individuals can also be involved in the project but they are not considered internal. In this section, these three roles will be analyzed in order to understand how they interact and which duties they have.

Product Owner

The product owner is the project's key stakeholder, and is the 'glue' of the whole project[73] since he interacts with development team, end users, Scrum Master, and everyone else involved. He represents firstly his own direct interests in the project and its results, but he should also defend the interests of the other individuals involved. His role is mainly to ensure that the project, when completed, creates value for the users.

The product owner is usually the decision maker about what features the product must have, so he should have some kind of business savvy, and be able to understand both the market, the supplier, and the end user. Moreover, product owner is supposed to have some management and tactical skills.

He is responsible for prioritizing the backlog during Scrum development: as seen, the backlog is a list of required features and functionalities to develop in order to complete the project.

[72] Lacey M., (2012). "The Scrum field guide", Addison-Wesley Publishing.
[73] Pichler R., (2010). "Agile Product Management with Scrum", Addison-Wesley Professional.

Product Backlog is needed to help the team to start from the top prioritized items to the lowest ones, in practice product owner ensures that the most valuable functionality is produced first and built upon. Product owner works with both the business and the team ensuring that everyone understands the tasks in the backlog, and he gives guidance to the team. It is important to underline that the product owner is not a project manager, because his focus is on ensuring that the team delivers the most value to the business.

Scrum Master

The Scrum Master is responsible for the Scrum process, he is the team's coach and helps the Scrum practitioners to achieve their goals. He is responsible for teaching Scrum to everyone involved in the project, so he coach the team, the product owner and the business to integrate the Scrum process. He is responsible for implementing Scrum and for ensuring that everyone follows Scrum rules and practices. By the way, he differs from traditional project manager because he does not assign tasks to individuals, nor he gives daily directions. He is there to allow the team members to focus on the project throughout every sprint.

Scrum Development Team

The Scrum Development Team is responsible for developing functionalities, and excel at sustainable development practices. Teams are self-managing, self-organizing, and cross-functional, and they are responsible for figuring out how to turn product backlog into an increment of functionality, as seen in the previous part about the Scrum process. Self-organized means that the team is not managed or directed by others, as it used to be in the traditional methods where management efforts were separated and centralized. Teams are cross-functional because every team has all the competencies and expertise needed to get the job done, each team member has different skills that are combined with these of other members.

In order to be more effective the team should be composed of 5 to 7 individuals, and each one should have different skill sets so that they can train the others, bringing added value to the project. The Scrum team operates as a single entity: team members are collectively responsible for the success of each iteration and of the project as a whole. This team organization increases flexibility, creativity, and productivity.

2.5.3 Benefits and advantages of Scrum

First of all, Scrum makes a clear distinction between people who are involved in the project and other individuals who might be just interested in the project, clearly the first ones are those who are responsible for it.

Everyone involved in a project developed using Scrum can benefit from it. The Scrum team benefits from clearly defined rules, and from interaction and improved communication between members. The management benefits from the fact that the team runs itself, since they are self-organizing entities who operates as a whole, making members feel more motivated and committed. Stakeholders benefit from using agile Scrum thanks to the eye continuously kept on the project, which allow them to control and keep it going in the right direction: in every sprint they know exactly what happens, are always updated and informed, and they can bring changes or add new requirements during the project development. Finally, also product owner has his advantages, because thanks to the product backlog and the prioritization of stories he obtains the desired result. This method allows to treat the most important aspects first, leaving for later less important ones.

By using agile Scrum, the work is divided into simple pieces. This helps everyone involved in the project to be more focused and to apply changes in every small step, thanks to lessons learned during the development and thanks to external feedback from people interacting with the product. Feedbacks are gathered more quickly and they help the team to improve the product.

Scrum forces people to think in terms of priorities[74] as seen in the section dedicated to the Scrum process, and this brings people to stay focused on generating results instead of blindly following a map or a plan.

Main advantages brought by Scrum management are set out below.

The first advantage is customer satisfaction[75]: as seen above, Scrum facilitates changing customer requirements thanks to the iteration process. This flexible way of working allows for changes in requirements and addition of new features that usually do not arise in the planning phase. Moreover, keeping the product backlog updated and always prioritized, allows the project to be responsive to changes. Changes are accepted and welcome, and there is no need

[74] Sutherland J., (2014). "The Art of doing twice the work in half the time", Crown Business.
[75] Layton M.C., (2015). "Scrum for Dummies", John Wiley and Sons.

for disrupting the project flow, since changes will be implemented in the following sprint. Customers are involved in the project throughout its entire realization, they collaborate as part of the team driving the product development towards the result they have in mind. This leads to a high customer satisfaction.

Likewise, team morale is improved, since teams are self-organized and decisions are not exclusively taken by managers; also team members have an important role when taking decisions. This improves team morale because members are more creative, innovative, and more focused. Indeed, the project realization is in their hands and they are responsible for the project success. Team's cross-functionality allows members to gain additional skills and grow thanks to mutual teaching, and face-to-face communication helps them to reduce frustration that might arise because of miscommunication. By dividing the work and focusing on small steps, team members feel less stressed and are more productive. In the end, if everyone is on the same level and the relation is peer-to-peer, there will not be organizational barriers, communication fears, or any other obstacle. As just mentioned, there will be an increased ownership since each team member is responsible for the final success and quality is ensured.

Another advantage is transparency: Scrum makes the work transparent, since it encourages everyone involved in the project (development team, product owner, and client) to communicate. Stakeholders can take informed decisions because they have, during the whole development phase, a clear idea of the project progress. The progress status is visible so that at any moment it is clear what the end product will be like. All the meetings provided by Scrum analyzed in the previous section are tools to increase transparency, facilitate communication, and give everyone a detailed idea about the accomplishment of the activity. In addition, these meetings allow measuring individual productivity, and early identifying and removing any issues for the future.

Scrum brings another advantage which deserves to be highlighted; increased quality and decreased risk. The risk of project failure is reduced thanks to the shorter delivery time and its organization in sprints. Prioritization of tasks in each sprint makes sure that higher-value and more risky requirements are developed and delivered first. By planning, testing, revising, informing all the stakeholders, and dividing the work, the project's quality is ensured. Scrum provides a framework which encourages constant feedback. Every single sprint is followed by a review meeting, which allows the team to think about the just completed one, and discuss

improvements for the next. This allows for adjustments according to the client's requirements and continuous acknowledgement of product features. Project control is increased because priorities are adjusted throughout the entire project, and thanks to the daily scrum meeting issues are faced as soon as they arise. Review meetings help to provide the team with product owners' feedback, which is necessary in order to ensure the quality of the project.

In addition, it has been noticed that Scrum **delivers value to the end customers** 30% **faster**[76] than traditional methods, because all the upfront documentation needed in the waterfall method is neither needed nor requested anymore. The focus on prioritized items makes sure that the most important requirements are delivered first, allowing the product to be competitive.

Unlike the other methodologies explained above, Scrum is a framework that focuses on improving practices aimed to manage the entire development process, to enhance business value, and reduce as much as possible resource wastes.

Since Scrum do not provide for mandatory technical practices, it lends itself to be used to improve and complement other methodologies that conversely do, such as the above mentioned ones, or an existing method already adopted in an organization as well. This last eventuality is analyzed in detail below.

2.6 A comparison between Agile and Stage-Gate

As seen in the previous chapter, traditional project management has disadvantages that cause inefficiencies and make the development cycle of a project very heavy, risky, and slow. Two more advanced methods came up to overcome the pitfalls of traditional methods,[77] respectively Stage-Gate for new physical product development projects, and Agile for software development projects.

The two systems have many clear differences. Stage-Gate is a macroplanning process, which provides for multiple, but limited in number, stages covering the entire product development project from its ideation to its launch, each one with a specific structure and a set of best

[76] Layton M.C., (2015). "Scrum for Dummies", John Wiley and Sons.
[77] Reagan B., (2012). "Going Agile: CA Technologies, Clarity PPM Division's transformative journey", Digital Celerity.

practices aimed to guide the process. Each stage is undertaken by a cross-functional project team, namely members from many different business sectors (sales, technicians, marketing, and so on). To summarize, Stage-Gate is a framework aimed to give instructions on which projects should be done and what has to be done in each project.

On the other side, Agile is a microplanning methodology, which provides for a number of short development cycles, from two to four weeks, each one worked by a dedicated technical team together with the customer. Agile was created precisely for software development projects, and it provides for a way to rapidly create a working software under a continuous customer approval. Each sprint should provide an incremental functionality that can be appreciated by project stakeholders. This means that it is not mandatory for each sprint to provide a market release, of course, but that each deliverable brings the product nearer and nearer to a deployable product. In this way, stakeholders are always updated on the project status, and are given the chance to adjust product requirements, that are never clear at the start of the project, or that simply change as the product goes on according to what they see.[78]

Differences between Stage-Gate and Agile are summarized in (Figure 11).

Figure 11 - Characteristics of Stage-Gate and Agile

	Stage-Gate	Agile
Type	Macroplanning	Microplanning, project management
Scope	Idea to launch	Development and testing, can be expanded to pre-development
Organization	Cross-functional team (R&D, marketing, sales, operations)	Technical team (software developers, engineers)
Decision model	Investment model—go/kill decisions involve a senior governance group	Tactical model—decisions about actions for next sprint made largely by self-managed team

Source: Cooper, 2016

Both methods benefit their users in many ways: Stage-Gate provides discipline, a framework structured in stages, gates where poor projects are terminated, a set of best practices, a focused team thanks to clear expectations, to mention a few. At the same time, Agile provides

[78] Boehm B., Turner R., (2004). "Balancing Agility and Discipline: A Guide for the Perplexed", Addison Wesley.

for an improved communication, an enhanced coordination, a faster product release, and a great flexibility to customer's change requests or technical challenges.[79]

At this point, a question spontaneously arises: can these two methodologies blend to create a new model getting the best from both and overcoming the disadvantages of both?

Real world practice answers 'yes, they can', with Agile-Stage-Gate hybrids.

2.6.1 Agile-Stage-Gate Hybrids

Benefits and promises of Agile have already been largely explained, and its specific creation for software development has already been stressed as well. Anyway, what emerged in recent times is that also manufacturers began to find Agile attractive, thanks to its adaptability and flexibility. Unfortunately, manufacturers had to recognize also that Agile in its pure formulation was not adaptable to new physical product development. Meanwhile, many large firms began to integrate the two methods also for managing IT projects. It is by responding to these issues that hybrid development processes mixing Agile and Stage-Gate arose.[80]

Physical product development is highly different from software development, at least under the point of view of work divisibility: software development is quite infinitely divisible, since a software consists of lines of code, while the same cannot be said for physical products such as food, jewelry, machinery, and so on.

Despite these diversities, physical product developers began to experiment Agile, integrating it into their Stage-Gate processes, modifying it to make it fit to their projects.

Integration between the two methods concerning a new product development process consists in adapting aspects of Agile to Stage-Gate process, in order to keep the best from both.

Firstly, it must be noticed that sprints are integrated, but are usually limited to the development and testing stages. Each sprint keeps the nature it has in software development projects, which is the processing and deliver of a working piece of product, but here it must be intended as the

[79] Begel A., Nagappan N., (2007). "Usage and perceptions of agile software development in an industrial context: An exploratory study" in *First International Symposium on Empirical Software Engineering and Metrics*, IEEE Computer Society.
[80] Karlström D., Runeson P., (2006). "Integrating Agile software development into Stage-Gate managed product development", Empirical Software Engineering (11), 203-225.

result of a physical manipulation. Thus, in this hybrid model, each sprint should end with a product version that, starting from a mere idea or concept, is nearer and nearer to a prototype that can be shown to the stakeholders, in order to obtain their feedback.[81]

Secondly, since the standard duration of sprints usually utilized in software development (from two to four weeks) is clearly too short to allow appreciable results for physical products, hybrid model provides for longer sprints: each organization chooses its appropriate sprint length according to many factors (product, resources, product life-cycle, customer's requirements, and so on) sometimes lasting for months.

Another adaptation concerns the frequency of meetings: while the short time-boxed sprint in software development provides for both a sprint planning meeting and a sprint review meeting with senior managers, long lasting iterations requires a lower frequency. A weekly one-day meeting between senior managers is enough to review and plan the project. Clearly, everything depends on iteration length: each firm must be able to choose a balanced meeting frequency according to its own needs.

Exactly the same as for software development, hybrid models provide for milestones, where a review of a physical predetermined objective is done. It is here that gates concept emerges: at any milestone, if a project does not meet all the planned requirements, it can be terminated.

Last but not least, another aspect of Agile that firms integrated into their Stage-Gate process is the limitation of bureaucracy and documentation, considered as a bulk of work bringing no value to the project. This lean approach to documents allows the team to be more focused on their work, so that the project go faster towards its completion. This aspect derives from one of the most meaningful principles of Agile: 'Simplicity is essential'.

What emerged from the research of Karlström and Runeson (2006) is that the integration of the two methods works, and this hybrid approach returns many positive outcomes.[82] Another research by Sommer et al. (2015) highlights almost the same advantages deriving from hybrids

[81] Cooper R.G., (2014). "What's next? After Stage-Gate", Research Technology Management 57(1).
[82] Cooper R., (2016). "Agile-Stage-Gate Hybrids", Research-Technology Management 59(1).

of Stage-Gate and Scrum method, the Agile version that emerged to be the most popular one among physical product development firms using Agile.[83]

Hence, advantages of Agile-Stage-Gate hybrids are the following:

- Enhanced team communication and coordination: agile daily meetings make the team communicate better, feeling the control of the project and consequently its progress.
- Improved attitudes: the previous benefit leads the team to feel more comfortable and motivated.
- Customer feedback allows flexibility: Agile helps project manager to understand customer's needs so that he can feel sure on features the product should have. This allows for a better product with a flexible design.
- Improved planning: early feedback from the customer, provided by Agile, on main product features leads a more efficient scheduling and avoids useless and dangerous delays on crucial requirements development, allowing for a more efficient project.
- Documentation is easier to manage: documents helps the development team in understanding priorities.
- According to the above mentioned researches, hybrid approach brings also some challenges to users:
- Team isolation: since teams are full-time employed in a single project, they tend to isolate from other parts of the organization, which may need their work as well. If the problem of finding dedicated team members was not enough.
- Long-range shortsightedness: although focus on the current sprint often represent an advantage, there is the risk of losing sight of the long-term planning and objective.
- Internal conflicts: managers in particular who inevitably must cease some control of the project during sprints might feel conflict among themselves.

[83] Sommer A.F., Hedegaard C., Dukovska-Popovska I., Steger-Jensen K., (2015). "Improved product development performance through Agile/Stage-Gate hybrids: The next-generation Stage-Gate process?", Research-Technology Management 58(1).

- Divergence of reward systems: Agile and Stage-Gate clearly imply different roles inside their structure. If the company has always used Stage-Gate, introduced agile practices might result not in line with the company reward system.

To conclude, the blend of the two methods allow Agile to generate efficiency, speed, and focus, while Stage-Gate offer a framework for coordination with other teams and communication with different management areas. An Agile-Stage-Gate hybrid works and brings positive results, and according to Cooper (2016) such an integration "may be the most exciting and significant change to the new-product process since the introduction of gating systems more than 30 years ago"[84].

2.7 Conclusions

This chapter aimed to outline the agile philosophy and the project management methodologies based on it. Agile was created to overcome the difficulties coming from the rigidities of traditional project management, which with its one-way stage-based approach did not fit to the need of flexibility of software development projects. In particular, Agile introduced some revolutionary concepts, namely communication and collaboration among everyone involved in the project, adaptation and evolution of the project itself, guaranteed by main characteristics such as frequent deliveries, flexibility to changes that may occur, and continuous stakeholders' feedbacks. The Agile Manifesto and Principles help software development teams in focusing on the practices that are really relevant for the project to succeed.

Many development methods based on Agile was created over time, each one with its diversities: Extreme Programming is characterized by simplicity and dynamism, Crystal by attention to project's criticality and complexity, Dynamic Systems Development Method by priority of fixed time and budget over number of functionalities developed, and Kanban by continuous improvement. Scrum has been analyzed in more depth thanks to its popularity and wide adoption all over the world. Scrum offers a framework and practices that promote transparency, collaboration, flexibility to project's changes during the development, motivation

[84] Cooper R., "Agile-Stage-Gate Hybrids", Research-Technology Management 59(1), 2016. P. 28.

and commitment of the entire project team, continuous feedbacks, and an iterative development planning.

Seen the advantages brought by using Agile, many manufacturers already using Stage-Gate for their product development projects, began to integrate and blend the two methods, giving life to Agile-Stage-Gate hybrids. Some adaptations of Agile have been made in order to fit to the physical product industry, but some researchers demonstrated that these hybrids work, returning many positive outcomes. In few words, Agile-Stage-Gate hybrids, even though presenting some challenges, it took the best sides of both methods generating efficiency, speed, focus, and frameworks for coordination and communication.

CHAPTER 3. AGILE CONTRACTS

3.1 General considerations on contracts

When working a project, the main priority of everyone involved is to deliver it successfully. What explained about Agile and Scrum practices in the previous chapter cannot be useful and effective without being supported by a legally structured agreement between parties specially designed for project carried out with that kind of methodologies.

The structure and legal aspects of contracts drew up for projects carried out with more traditional development styles remain the same for agile contracts. "The key difference is the approach to and understanding of *operational process* and *delivery* and how this is captured in or intersects with contracts. […] An agile approach enables rapid incrementally deployable deliverables and collaborative decision-making between the parties, and so relieves pressure on liability, warranty, and similar issues."[85]

Software development projects are commonly considered by lawyers as akin to construction projects, with all the uncertainties of what they imply: long time before something usable is delivered, very late client's feedback, long payment cycles, huge problems in case of early abandonment of the project. Agile development annul all these assumptions. [86]

As seen in the previous chapter, the third statement of Agile Manifesto is *Customer collaboration over contract negotiation.* Before continuing, it is important to remember that it is not a dichotomy: in fact it does not mean that contract negotiation is ignored in favor of collaboration, but only that a collaborative behavior between customer and supplier during project development is crucial for its successful delivery. For this reason, the contract should be drafted in a structure that encourages collaborative practices, in order to allow legal professionals to go along with the goal of agility towards project's success.

The traditional non-agile way of contracting provides that an early and detailed specification of requirements is produced and approved before starting the project and that the final product

[85] Larman C., Vodde B., (2010). "Practices for Scaling Lean & Agile Development: Large, Multisite, & Offshore Product Development with Large-Scale Scrum", Addison Wesley.
[86] Arbogast T., Larman C., Vodde B., (2012). "Agile Contracts Primer", Agile Contracts.

is delivered at the very end of the project. This means that the customer cannot see and check the final product until that moment, running the risk of realizing too late of any inconsistency between the final product and his expectations. It is understandable the huge pressure on negotiating crucial contract terms such as liability and warranty, which for this reason should provide maximum protection to the customer.

In the same way, also the supplier can feel a pressure deriving from a final delivery, because it may not be able to evaluate the overall effort for working the project until the very end.

With agile method these kinds of pressure are significantly relieved because, being focused on delivering real value instead of stacks of documents, it provides for the early and frequent delivery of working parts of the project, adding gradually more functionality. Usually deliveries of done parts of the project are planned every two weeks, but a different timing can be established by mutual agreement of customer and supplier. This regular, constant and shared system of deliveries radically changes the way of negotiating terms, in contrast to problems and difficulties of negotiations due to the certainty (and fear) that it had to pass a lot of time before the final, complete delivery.

The constant delivery approach solve crucial issues related to liability and exposure: the customer is sure he is paying for something he knows and wants, while the supplier can be sure he is delivering a product in line with customer's expectations and from which he can recognize revenue. In this way, if the relationship between customer and supplier for any reason ends before the project is finished, damages are contained. The customer will not have paid for something he expected and he did not obtain, while the supplier will not have worked and lost time on something he is not going to be paid for. It is clear this is not a desirable situation because both parties' expectations are not satisfied, but damages in terms of business and exposure[87] are significantly reduced compared to what it may be in a traditional waterfall project scenario.

This leads to a consideration about incentives and penalties. In traditional contracts it is common to find incentive and penalty clauses, in the belief that they are useful to increase commitment and work quality. However, evidence shows exactly the opposite: incentives drive people involved in the business relationship to dysfunctions, such as quality and transparency

[87] For practical reasons, here is not addressed the issue of expectation costs, consequential damages, lost profits, and other damages.

decrease, trust deterioration, gaming increase, and others. Penalties, here intended as penalties related to performance variation, lead to the same problems. Incentives and penalties encourage negative competition between parties instead of a positive cooperation. How to prevent such behaviors? Here the answer is the simplest that comes to mind: simply avoid performance-based incentives and penalties in contracts. In fact, in agile contracts, in case of customer's absolute discontent with supplier performance, he can decide to conclude the commitment at the end of any iteration.

3.2 Common subjects of agile contracts

It is now clear that before drafting an agile contract it is crucial to deeply understand agile method principles, the lean development it implies, and consequently its impact on the entire business relationship regulation between parties.

Every contract must be drafted specifically for the project it has to regulate, working carefully on each topic. A list of the main topics is given below:

- "Delivery cycle
- Project scope
- Change control
- Termination
- Acceptance
- Deliverables
- Timing of payment
- Pricing
- Warranty
- Limitations of liability"[88]

Topics such as acceptance and termination of a project are similar for agile and traditional project contracts: the difference lies in the elements contained in them, which for agile-project contracts foster collaboration, learning and evolution. Below is an extended analysis of each

[88] Larman C., Vodde B., (2010). "Practices for Scaling Lean & Agile Development: Large, Multisite, & Offshore Product Development with Large-Scale Scrum", Addison Wesley. P. 499 e ss.

topic, concentrating on the impact of Agile Manifesto's two principles "*Responding to change over following a plan*" and "*customer collaboration over contract negotiation*" on them.

Delivery cycle

From the beginning of the project, the delivery cycle is always the same and constant. It provides that at the end of each iteration a "piece" of deployable product is delivered, and it has to add some improvement. It means it may present new features that are not useful and deployable in themselves, but contribute to bring the product closer to an interesting deliver.

Incremental delivery is not a trait distinctive of agile contracts, as it has always been used also in traditional development contract identifying intermediate milestones set by objectives or date with determined acceptance criteria or statements of work.

What really makes the difference about delivery cycle and milestones in agile contracts is a set of three distinctive characteristics: d*oneness and deployability*, *duration*, and *timeboxing*. Scrum defines the *doneness* of the product increment in terms of activities carried out in order to provide a potentially usable (*deployable*) additional piece of product. The rules to consider a thing done are defined by customer and supplier together at the beginning of each iteration. *Duration* is the length of the iteration, and *timeboxing* is the concept of fixed time but variable scope.

Project Scope

In Agile, contracts it is not determined a precise and invariable project scope, since variations of scope specificity are allowed. Changes are usually related to pricing. On the one hand there are target-cost contracts, where the planning of the project scope at the beginning of the project in order to establish the overall cost, on the other hand there are progressive contracts, where project scope is defined for each iteration.

Change management

Agile approach, encouraging change and adaptation, can represent an issue when contracting because of a re-prioritizable backlog and an iterative project planning. In such a situation, it is

not possible to foresee all the possible changes that may occur and put in the contract, indeed in agile contracts, concepts related to changes can fall under two categories:

- Change in relationships between parties. This case is managed in very much like in common contracts, keeping in mind that in agile approach such changes are less dangerous thanks to iterative deliverables and related payment.
- Change in project scope. This field must be carefully managed in order to avoid going against agile building blocks: easy and frequent changes must be allowed without imposing rigid and heavy procedures. Anyway, the degree of flexibility to changes can be regulated in advance.

Termination

In contrast to the conventional concept of natural end of the project consisting in finishing the product, in agile approach an early termination of the project should be seen as a positive event, since it does not necessarily mean failure but it can mean that the desired result has been achieved quickly. For this reason, in compliance with agile philosophy, the customer should be allowed to stop the project at the end of any iteration without any penalty. To overcome expensive problems for the supplier due to a commitment interrupted unexpectedly much earlier by the customer, a set of termination clauses consisting of a sliding scale of penalties can be included.

Acceptance

This is a very delicate issue, since it can represent a source of conflicts between parties. Since agile approach provides for continuous and incremental deliveries, also customer acceptance procedures follows this path: each iteration should end with his acceptance. This procedure is simplified by the planning done at the beginning of each iteration, which defines what should be achieved in its end. Final acceptance is the sum of all the smaller previous iteration acceptances that occurred during the entire project life cycle. What is crucial here is to define a contractually clear *framework for acceptance*, in order to consolidate all the results obtained in each iteration on a shared agreement between parties. Such clauses can define all the elements aimed to clarify the definition of done and accepted work.

Limitation of liability

In contract negotiations, limitation of liability clauses has always represented probably the most difficult issue, but Agile can help it since the fact that each iteration ends with a continuous usable delivery can reduce the pressure on liability. Indeed, in this way defects that may occur are early detected, so that negative consequences have a limited impact on operations before being corrected.

Warranty

In the same way, agile approach attenuates also issues related to warranty thanks to the confidence and lesser risk brought by incremental acceptances. Each iteration should end with a customer's greenlight also on warranty aspects, in order to build, iteration by iteration, an overall grounded warranty to the final product.

Deliverables

Usually contracts include a detailed and rigid list of what is to deliver, and what requirements these deliverables should meet in order to be accepted. In agile approach this rigidity should be avoided for many reasons. Firstly, it shift the focus on waste activities rather than on working software. Secondly, the focus should be on collaborating to create a working software, rather than wasting time in negotiating and adapting at all costs to established quality requirements. Thirdly, agile approach encourage a continuous learning and response to changes, and the project should follow the path, rather than trying to impose rigidly programmed plans. Last but not least, rigidity do not allow an evolutionary and creative work, since it leads to wrongly believe that a precise prediction brings to precise and punctual deliveries.

Anyway, it is much easier to understand what are valuable deliveries when they are frequent and incremental, rather than listed in the contract.

Timing of payment

In Agile, the more intuitive and common system is to pay in the end of each iteration, when the related deliverable acceptance occur. Clearly, this issue can be more easily managed when projects are simple, where the total amount of the iteration is paid. In more complex projects

payment schemes are in turn more complex, and can be build in many different ways. An example can be a payment scheme that besides iteration payments envisages a final payment when the project ends.

Pricing

1. Time and materials

Variation of time and materials is a simple and recommended pricing model, that can be applied to contracts where the scope is fixed or flexible. The traditional issues are mainly two: customers pay a lot before they can see results, and they can doubt to get the right value for money. These issue are attenuated in agile approach thanks to high transparency, progress measured in terms of done work, and the chance to terminate the project at the end of each iteration. Some devices are used to modify T&M in order to adjust risks from both parties: capped variations per iteration, per project or release, or per iteration with some adjustments. This last one foresees the possibility to adjust agreed terms in order to benefit the supplier in case of a higher-than-average productivity: if iteration end earlier than expected with an accepted deliverable, the supplier gain one half of the difference between the effective expenses and the cap, which are savings.

2. Fixed price per iteration (per unit of time)

This model is simple and easy to implement thanks to its predictability. It is commonly used in agile contracts in two main forms: the first is the definition and agreement of requirements before the iteration starts, that is almost the same as a fixed price for an entire project, but the smaller scope of each iteration make it easier for suppliers a clear and precise work estimation that prevent money losses. The supplier adds a fee to prevent risks related to variations in R&D work.

The second form contemplates highly flexible requirements or not defined at all, in which the customer confidence and trust in the supplier is a crucial element. Agile helps it with transparency, frequent deliveries, and the possibility of early project termination.

3. Fixed price per unit of work

In traditional development, a unit of work was a document, a prototype, or another separate piece of the entire project, while in agile approach the unit of work is related to something that effectively works, which is working software features. A unit of work can be estimated

according to the points of stories, or features, or function, or others. Since in Scrum a "point" refers to an estimated effort, this can be the basis to establish a cost. The price per point is determined following two main alternative schemes: an average based on previous projects, or a customized amount. When the price is customized, the customer pays an estimated average amount of points for the first few iterations, while accurate data are collected in order to adjust the fixed price per point in agreement with the customer. A profit margin will be added to the price.

This model is compliant with agile approach, since it is oriented on using the working software as a measure for progress, it is value oriented, it allows changes, and so on.

What is crucial here is to find a clear and shared framework between customer and supplier on what can be considered a point, since it has not been given an unambiguous definition yet.

4. Pay-per-use models

This model consists in the customer paying for every use of a pre-built or customized system that is deployed for him, in practice the bill is based on frequency of use. This method tends to match customer and supplier interests, and the system works if it is increasingly used. In case of pre-built solutions, the customer does not have maintenance costs, and he pays only for the extra customizations. In case of custom-built solution, the paying model can be one of the above explained or an adapted one.

5. Hybrid shared pain/gain models

Many hybrid pricing schemes exist, indeed they are adaptations of the previously explained one. A model applicable to Agile was proposed[89]: a discounted price based on units of work and in addition a discounted time and materials, which in agile approach it is translated in a discounted price per person-day and a discounted price per point. In this way if the estimated and effective effort do not coincide, which is whether the actual effort exceeds or is minor than estimations, the pain or gain of the project is shared between customer and supplier.

[89] Martin R., (2004). "Estimating Costs Up Front", Extreme Programming mailing list.

3.3 Contract models

This section is aimed to give an overview of the most common contract models usually considered in agile projects.

Fixed-price and fixed-scope contracts

This model tends to make both customer and supplier unhappy: the first does not get what he really needed or wanted, and the second easily loses money. Because of the constraints of price and scope, the supplier might lower the quality of its work, leading to an increase in customer costs to later remedies. For obvious reason, this model can create dynamics that lead to a loss in transparency and an increased gaming. In addition, since it does not allow for changes in scope, usually the customer do not get what he really need, so the supplier can increase its revenues thanks to change requests that are additional costs paid beyond the fixed cost.[90]

Anyway, despite all the negative aspects of the model, there are still many companies trying to make it work. Some keys have been identified in order to limit issues when using this model with agile approach. Firstly, an early requirement analysis must be accurate and detailed, with the definition of acceptance tests, and a precise assessment of all requirements, everything done by experienced people. Also prioritization of requirements should be attentively established. Secondly, scope changes must not be allowed: just in case, a new requirement can substitute an existing one, but only if the effort is almost the same. Thirdly, given the risk of this model related to the variability of software development, the overall price of the project should be increased to guarantee a higher margin to the supplier. A mention is also deserved by a recommended early termination clause: since a satisfied customer can early terminate the project, if it happens the supplier should be paid for a percentage of the unbilled value (e.g. 20%). Last but not least, experienced people with great technical skills should be involved.

Usually, payment timing for this model is per iteration, with a final amount at the end of the project.

To sum up, the fixed-price and fixed-scope model should be avoided by companies using agile approach, since it is more suitable for waterfall projects, but some attention to the above mentioned details can solve some main issues.

[90] Opelt A., Gloger B., Pfarl W., Mittermayr R., (2013). "Agile Contracts. Creating and Managing Successful Projects with Scrum", Wiley.

This kind of contract can be useful to build the trust and confidence between parties needed to follow up with other contract models.

Variable-price and variable-scope progressive contracts

Since in this model project scope is completely variable and defined for each iteration, while is defined only an overall pricing scheme, it is suitable for agile projects.[91] Pricing for each iteration is decided in agreement, and can be one of the above mentioned, and some variations of this model provide for a price cap, but anyway the customer is protected thanks to his right to terminate the project at the end of each iteration.

The parties together can create, before the contract is created, a non-binding *Release Backlog*, where the release goals are defined. This backlog is not binding because it is created only for a common understanding of the project, and for a more precise estimation of effort needed and the best pricing scheme.

It can easily be understood how many variations can be created playing with flexibility on the contract elements, depending on the level of trust, confidence, and collaboration between the parties, for example: capped-price (overall project price) and variable-scope, capped-price and partial-fixed-scope, and fixed-price and variable-scope. What is crucial when writing such a flexible contract is trust between customer and supplier: with a low trust, each party can protect itself with a series of short-duration "testing" contract, aimed to build a trust that can make the parties able and confident to shift to another kind of contract, for example a progressive one.

Target-cost contracts

Since contracts in themselves are not a source of trust, rather they more easily are a source of gaming to shift pains on the other party and obtain gains, this model helps in obtaining transparency and honesty thanks to the alignment of motivations of both parties. To well establish a target cost, some phases must be followed. First of all, customer and supplier collaborate to identify the overall project scope and all its requirements. Secondly, they estimate the cost of change in requirements and scope during the project development. With these elements the target cost can be established, from which the target profit will be calculated (for example 12% of target cost). In the end, all details must be shared with the

[91] Poppendieck M., (2005). "Agile contracts", Agile Conference Workshop.

customer. The characterizing element of target-cost contracts is a formula aimed to share pain or gain due to differences between target and actual cost. The formula for sharing unexpected costs or profits is defined by customer and supplier when writing the contract.
To sum up, since in this contract model responsibilities are shared, it encourages to improve collaboration and transparency in order to reduce wastes, and push both parties to do their best to gain the highest profit possible.

3.4 Conclusions

In the first part of the chapter, it has been shown that when managing a project with agile methodologies it is crucial to draw up appropriate contracts, able to follow the values and characteristics of this approach. Before drafting an agile contract, it is crucial to deeply understand agile method principles, the lean development it implies, and consequently its impact on the entire business relationship regulation between parties. The topics to take into consideration and to work on carefully when drafting a contract have been analyzed in order to give the basis to explain some main contract models.
As seen, many contract models already exist, but many possible adaptations of existing contract models are possible as well for customers and suppliers who want to find the best way to manage their agile project. The aspects to keep into considerations when writing a contract are many and complex, and are subject to mistakes. But in any way a contract is framed, a key considerations must be kept into account: trust, collaboration, and transparency, if present lead more likely to a successful and profitable project for both parties.

CHAPTER 4. AN AGILE CASE STUDY

In the present chapter a practical situation in which Agile has been applied to a real work environment is going to be analyzed. Object of my study is the Agile based business relationship between two companies, the service supplier called Eurecna committed to developing a software for the client called Open Software. The main and most interesting part of the chapter is the explanation of the Agile-coherent boundary object used in this business relationship to manage the project. Presentations of both companies are outlined below.

4.1 Eurecna

Eurecna is an independent private company based in Venice and operating in five continents, with a 20 year international experience on large projects in the economic, social and public sector, ICT consulting and services, and renewable energies.

Eurecna turnover has increased in the last three years: € 3.9 million in 2013, € 7.5 million in 2014, to end with a turnover of almost € 10 million in 2015. Eurecna EBITDA followed the trend, going from € 352 thousand in 2013, to €717 thousand in 2014, to € 796 thousand in 2015. Net profit has been volatile, since in 2013 it amounted to € 9,500, in 2014 to € 146,600, and in 2015 to € 75,800. The overall profitability of the company has not been brilliant, since ROE was 0.20% in 2013, 3.06% in 2014 and 1.56% in 2015. However, this decreasing of ROE is not due to ROA, which increased over the three-year period: this means that the company's assets returned positive results. Also ROI, the return on the company's investments, increased. What affects ROE is the Debt Ratio, which slightly increased in the three-year period.[92]

The company counts about 30 employees and is divided in 3 business units: Technical Assistance, Information & Communication Technology, and Renewable energy.

[92] Data collected from Aida.

Technical Assistance: Eurecna entered the world of technical assistance working in the field of private sector development in Central and Eastern Europe. Gradually, its areas of intervention expanded to regional, local and social development and, with the first steps of the EU enlargement process, Eurecna started to operate in the fields of public administration reform, EU integration and legal reforms. Today, Eurecna's TA activities cover five continents and fall under two main units: Economic and Social Development and Public Sector and Governance, but TA also works in the field of ICT, energy and environment in collaboration with ICT and Renewable Energy units.

ICT: this unit provides concrete and innovative internet solutions for private customers and for Technical Assistance. ICT initiatives require expertise, well-tested methodologies and best practices, holistic approach, multidisciplinary know-how and flexibility with structured approach during implementation.

To face these needs on the local and international market, Eurecna at the end of 2007, has completed a merger with a dynamic and fast growing Internet Company to set up a new Business Line focused on eBusiness and eGovernment solutions.

Renewable Energy Unit: is operating in an advisory capacity for a wide range of renewable energy technologies and is directly investing in the fields of wind power, cogeneration from solid biomass, production of liquid biomass, SHPP (Small Hydropower Plant) and waste-to-energy. Eurecna is currently managing a pipeline of renewable energy projects for an overall capacity of over 300 MW in Italy and almost 600 MW overseas – most of them in Romania – including the development of CDM projects in Macedonia, Montenegro and Morocco.

4.2 Open Software

Established in 1995 in Mirano (Venice), the company is specialized in designing, developing and providing IT solutions to county police headquarters.

Open Software turnover kept a positive trend in the last three years, going from € 4.2 million in 2013, to € 7.4 million in 2014, to end with € 8.2 million in 2015. Also EBITDA followed the trend, ending 2015 with € 744 thousand. Net profit of 2013 was € 23 thousand, but saw a

significant increase in 2014 and 2015 amounting to € 140 thousand on average each year. The overall profitability of the company significantly increased over the three-year period, indeed ROE stabilized at 22.08% in 2014 and 21.24% in 2015, 17% on average higher than 2013. The increasing of ROE has been accompanied by an almost twofold increase of ROA (from 3.63% in 2013 to 6.02% in 2015). ROI kept almost stable over the three-year period, on an average of 12%, and the Debt Ratio kept constant.[93] All the analysed data prove a good health of the company.

Open Software is the only company in Italy whose core business is the specific sector of local police. In order to provide to its customers more and more useful and innovative services, besides Visual Polcity, the management tool for local police activities, which will be explained in the next paragraph, Open Software realized www.PoliziaMunicipale.it portal, the most popular portal website among workers in the sector. This important editorial experience continued with the introduction of UfficioStudi.net, representing for more than a decade the best professional updating service able to ensure a targeted information and an effective help in the practical-operative activity of county police operators. Open Software is also partner with *Poste Italiane* and *Il Sole 24 Ore*.

As said, Open Software's services offer addresses to local authorities, provinces and regions. In order to meet market and customers' needs in an effective and efficient way, Open Software divided its structure in Strategic Business Areas.

Software Area: is committed in IT solutions development. Software are implemented using the more recent technologies available on the market. This area includes also assistance services and consultation on provided applications.

Service Area: data entry, data optic storage with 'cloud storage' systems, dedicated web services (for example 'info point' services for citizens) and printing services.

Hardware Area: supply of personalized hardware solutions.

Training Area: event organization and specific training.

[93] Data collected from Aida.

4.3 The collaboration between Eurecna and Open Software

In 1995, the year of Open Software establishment, the company developed its main product, Visual Polcity XP2, an IT solution consisting of a management tool for all the activities of local police. Visual Polcity allows administration of fines issued for breaking traffic regulations as well as commercial regulations, which are penalties related to businesses. Administration of fines is a very complex activity, since it implies the management of all the issues originating from them, namely: infractions report data entry, fine notification and shipping, fine payment with many different payment systems (also instalments), DMV information in case of missing data on the vehicle, import and management of penalties detected and collected by speed cameras and restricted-traffic zones infraction cameras, penalty points on driving licence, registration of persons involved, appeals management, accidents, to name but a few. Visual Polcity has been developed to support local police in managing all these activities with the simplest possible and intuitive procedures, within a system presenting the highest possible degree of automation.

Visual Polcity has always had a very important success: to date, Open Software counts over 478 county police headquarters across the entire national territory using Visual Polcity, and 254 among them adopted also the management in outsourcing of services related to sanctioning system (data entry, data optic storage with 'cloud storage' systems, dedicated web services and printing services).

Although Visual Polcity has always been a good and appreciated product, and the proof is in its diffusion over the national territory, it is also true that Visual Polcity was developed in 1995 by means of the time. In twenty years technologies have evolved, requirements have changed, legislation has varied, and Open Software has always adapted and adjusted the original software over time using different team members, causing a tangled software structure. The deciding factor that drove Open Software to change, is that Terms of Reference issued by local authorities began to request technical specification Open Software was not able to meet.

The collaboration between Eurecna and Open Software, therefore, originates from the wish and the need of Open Software for a radical makeover of Visual Polcity, in order to improve the usability, efficiency, attractiveness, and quality of its main IT solution, representing its core business.

Open Software decided to rely upon an external organization to develop an innovative solution, and Eurecna was chosen for this initiative thanks to its expertise in the technological solutions field.

When deciding the method to use for the software development, the parties analyzed the situation: the product owner actually did not know at the very start of the project exactly all the features he wanted in the new software, rather he had an overall idea of requirements. This means that the project would have evolved through its entire lifecycle, defining gradually every feature as the product grew. In this scenario, the parties agreed on the procedure to adopt, and established they would have used Agile, because it represented the best solution for such a software development project: very complex, in continuous evolution, and subject to frequent changes.

In order to better organize the work for the development of new Visual Polcity software, it was necessary that both parties together chose an appropriate tool for boundary objects management to communicate effectively about the project.

4.3.1 Boundary objects

The concept of *boundary objects* came up the first time in 1989 when Star & Griesemer defined them as "those scientific objects which both inhabit several intersecting social worlds and satisfy the informational requirements of each of them. Boundary objects are objects which are both plastic enough to adapt to local needs and the constraints of several parties employing them, yet robust enough to maintain a common identity across sites." [94] Thus, according to the two authors, boundary objects are means that provide a point of contact for communication between people with different social and cultural backgrounds, and thanks to their dual nature of flexibility and rigidity, they are able to adapt to the different social worlds' needs maintaining at once their structural identity across actors. For this reason, since they represent a mean of communication recognizable by several worlds, the correct creation and management of

[94] Star S.L., Griesemer J.R., (1989). "Institutional Ecology, `Translations' and Boundary Objects: Amateurs and Professionals in Berkeley's Museum of Vertebrate Zoology, 1907-39", Social Studies of Science, 19(3), 387–420. P. 393.

boundary objects is crucial for building and maintaining social and relational cohesion among actors belonging to different backgrounds.[95]

Usually, boundary objects are considered as static artifacts useful to help development, rather than adaptable and flexible tools that help in building a shared understanding and information across boundaries.[96] However, what can be considered boundary objects under an agile development perspective, are all these documents and deliverables that facilitate communication between all the team members, aimed at a shared understanding of flows, interactions, and functionalities, such as sketches, user stories, design specifications, and prototypes.[97] In few words, boundary objects are all those interfaces able to build communication between organizations or team members[98], those objects with which the actors interact.[99] Artifacts are an integral part of work processes, since they help team members to carry out tasks, both in a technical way and participating in building the social dynamics of organizations.[100]

An example of a frequently used boundary objects when developing in Agile, are collaboration software for planning, designing, and testing websites, web applications, and enterprise software. These tools, which in practice are functioning and interactive prototyping tools, allow users to easy create clickable wireframes, mockups, and simplified prototypes of user interfaces. Once a prototype is created, it can be shared with team members and project stakeholders for real-time reviewing and testing tool. This kind of tool is used in the design phases of software development, when coding has yet to begin, and involve end users, product owner, and developers.

As can be easily understood, boundary objects represent a vital element in project development, but in order to be really effective, a boundary object should not be excessively time consuming to produce and bring to deliver. As found by Persson (2014), spending too

[95] Barrett M., Oborn E., (2010). "Boundary Object Use in Cross-Cultural Software Development Teams", Human Relations, 63(8) 1199-1221.
[96] Ohst D., Welle M., Kelter U., (2003). "Difference tools for analysis and design documents", International Conference on Software Maintenance, IEEE.
[97] Persson J., (2014). "Communication through boundary objects in distributed agile teams", Linkopings Universiteit.
[98] Zaitsev A., Gal U., Tan B., (2014). "Boundary Objects and Change in Agile Projects", Australasian Conference on Information Systems, ACIS.
[99] Mall R., (2004). "Fundamentals of Software Engineering", Prentice Hall of India.
[100] Bechky B.A., (2003). "Workplace Artifacts as Representations of Occupational Jurisdiction", The American Journal of Sociology, 109 (3) 720-752.

much time on producing documentations very often results in heavy and complex material that is difficult to manage: the necessity of giving detailed directions on product design must be supported by a middle ground between level of detail and time spent to produce them. Simplicity helps facilitate interpretation by developers, who should not lose their time trying to understand design specifications (for instance) with the risk of misunderstandings and consequent useless and costly mistakes.

As previously said, Agile encourage communication and human interactions rather than documentation. But when the team members are scattered, communication becomes inevitably more difficult: in this scenario, boundary objects became even more important as communication and collaboration tools.[101] At the same time, since Agile encourage changes, as one of the four main values state, also boundary objects are influenced by continuous changes: this can consume their capability to enhance collaboration and communication due to mismatches in understanding and consequent decrease in team performance.[102] A hidden effect is that organizations, although using Agile, inhibit changes in projects and consequently on boundary objects in order to avoid the risk of losses, but risking to reach an impasse. Since this can represent a serious issue for agile development, a deep understanding of the way project changes affect the efficiency of boundary objects is crucial in order to choose the more suitable objects and better manage them. Two important precautions should be taken: firstly, boundary objects must be maintained as flexible tools, and secondly be sure that objects do not contain too much information or try to connect too many different backgrounds. If objects lose their flexibility and assume too many meanings becoming confusing, users can be led to abandon them altering the communication practices.

The best way to share all the material and information about the project is to put it on a common virtual environment, reachable by all the team members, namely a project database or a product backlog. These environments separately are mere container of data and information, but can be seen also themselves as boundary objects. These tools are called *repositories* for a boundary object type[103], and it is clear how distributed agile teams have a

[101] Barrett M., Oborn E., (2010). "Boundary object use in cross-cultural software development teams", Human Relations 63(8), 1199–1221.
[102] Cheng L., Subrahmanian E., Westerberg A.W., (2003). "Design and planning under uncertainty: issues on problem formulation and solution", Computers and Chemical Engineering 27 (6), 781-801.
[103] Star S.L., Griesemer J.R., (1989). "Institutional Ecology, `Translations' and Boundary Objects: Amateurs and Professionals in Berkeley's Museum of Vertebrate Zoology, 1907-39", Social Studies of Science, 19(3), 387–420.

strong need for a functional and effective repository, that implies also easy access to information, to carry out their activities. The problem they present is disorganization and no effectiveness. It is from the need of organizing all the information in a common ground that arose tools helping in organizing it. These tools represent a shared environment where all the team members collaborate to organize their work. User stories are put in and prioritized in the backlog in order to make everyone aware of which is the next part of the project where it is needed to focus on.

When the product owner has to communicate developers what deliverables he expect and how the work must be done, user stories come to attention as deliverables themselves. User stories, with their standardized form, are an efficient communication mean[104]: they are short sentences indicating the perspective (user) and the goal (what the user want to do). This boundary object offer some benefits. Firstly, user stories as deliverables allow the empirical process control by stakeholders, which is the basis of Scrum method, based on a mandatory three-phase path: visibility, inspection, and adaptation. Secondly, stories are usually created by people who know deeply users' needs and goals, therefore stories can be seen as detailed explanations of features to develop in order to respond to users' needs.[105] Thirdly, user stories help in connecting categories of users and their respective goals: this helps in understanding the overall project direction and developing a suitable design, even producing sketches or prototypes.

Tools helping in organizing user stories or other boundary objects can be considered as strategy tools[106], intended as tools supporting managers in decision making and helping them in crafting organizations' competitive strategies.[107] Tools belonging to this category are numerous and diverse, but the one Eurecna and Open Software chose for their collaboration is Pivotal Tracker, which is going to be treated in depth in the next paragraph.

[104] Jurca G., Hellmann T.D., Maurer F., (2015). "Integrating Agile and User-Centered Design - A Systematic Mapping and Review of Evaluation and Validation Studies of Agile-UX", Department of Computer Science, University of Calgary, Canada.
[105] Bayer H., Holzblatt K., Baker L., (2004). "An agile user-centered method: Rapid Contextual Design". Extreme Programming and Agile methods – XP/Agile Universe 2004, Lecture notes in Computer Science, 3134, 50-59, Springer.
[106] Spee A.P., Jarzabkowski P., (2009). "Strategy tools as boundary objects", Strategic Organization 7(2), 223-232, Sage Publications.
[107] Clark D.N., (1997). "Strategic Management Tool Usage: A Comparative Study", Strategic Change 6(7): 417–27.

4.4 An Agile boundary objects management tool – Pivotal Tracker

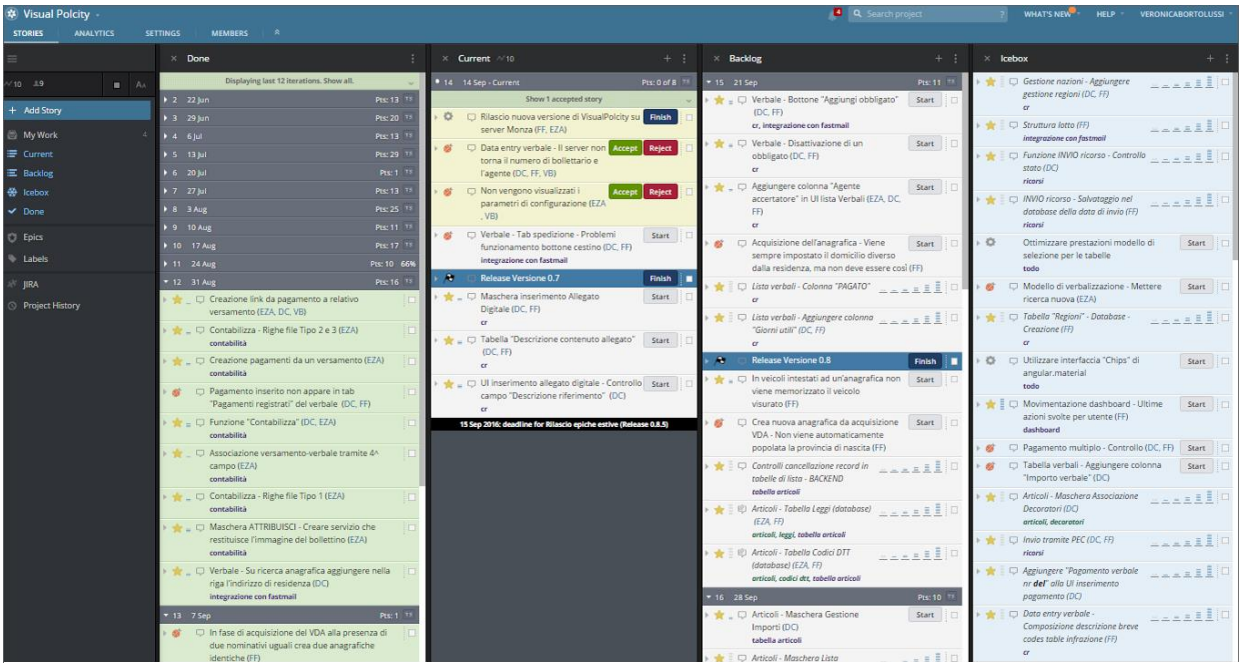
Pivotal Tracker is an agile-style project-planning tool useful for software development teams. This software helps project managers, project owners, and development teams in organizing work, tracking progress, and forming realistic expectations about when work might be completed based on the team's ongoing performance. Pivotal Tracker encourages a practical agile software development process.

When projects involve more than two people, keeping track of what needs to be done begins to represent a problem very quickly as problems arise and get handed out to different people. In this scenario keeping track of when, by who, and whether each issue is handled can not be done by using traditional communication channels as emails, meetings, and phone calls.

It is here that stands out the importance of a tool like Pivotal Tracker, that in addition to the functional aspect, it is also characterized by a very intuitive and simple usability. This software has been designed in such a way that no extra mental effort is required to deal with using it, in order to be used easily by the entire project team without any problems. It is not needed to be a tech or an engineer to properly and effectively use it, and this make the difference between project tracking solutions when choosing which one to use for your project.

Pivotal Tracker provides a clean and easily understood user interface design, structured in four main panels: done, current, backlog, and icebox (Figure12).

Figure 12 - The four main panels of Pivotal Tracker user interface



Source: pivotaltracker.com

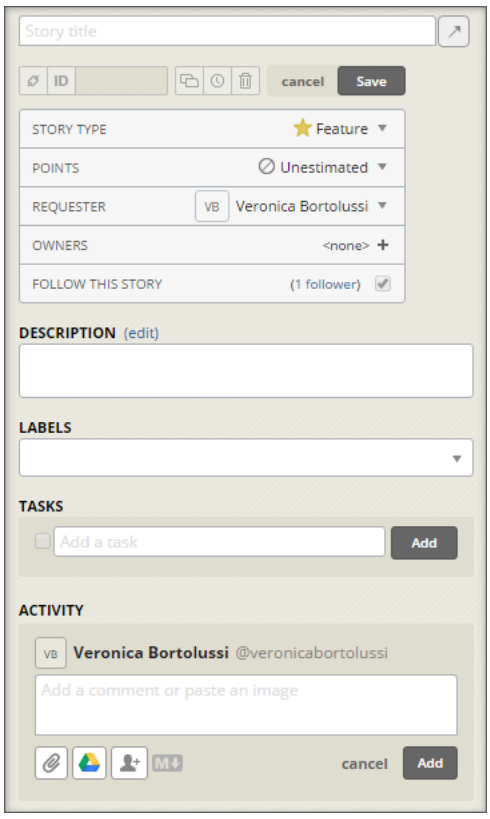
In order to let the reader understand how this tool works, the workflow is set out below.

At the start of a project, before any coding begins, the product owner, developers and the project manager collect features and requirements at a general level of detail. Ideas are put down in a planning session, where discussion is important to bring to a clear and shared idea of the overall project, including deliverables and scope.

4.4.1 Writing stories

Four different types of *stories* exist in Pivotal Tracker: features (or user stories), bugs, chores and releases. The framework for writing a story is the same (Figure 13), but since each one is different from the others for many reasons, a further explanation is given below.

Figure 13 - Framework for writing a story



Source: pivotaltracker.com

Features

Pivotal Tracker helps in organizing projects in the form of stories. Stories are the basic unit of work used in this tool, and different teams across projects and companies have different styles of story writing, depending on what is most effective given the goals and context of the project. However, even if the best and more effective way is difficult to determine, there are some core principles that guide story writing.

Focusing on the structure, a story should be the smallest incremental feature that can be added to a product that delivers user value. Stories should be short, simple descriptions of a feature told from the perspective of the person who desires the new capability, usually a user or customer of the system. They typically follow a simple template: *As a <type of user>, I want <some goal> so that <some reason>*.

Some principles allow the project manager to understand if the story has been properly written or not: it is the SMART goal setting, a methodology giving criteria to guide in the setting of objectives. SMART helps in creating verifiable trajectories towards a certain goal in several fields of management. In this case, SMART is going to help in understanding if stories have been created in a proper way. SMART stands for:

- Specific. The goal of the story should target a specific area of improvement or answer a specific need. To set a specific goal, the story should answer to the five W questions: What do I want to accomplish? Where? When? Who is involved? Why do I want to achieve this goal? Which are requirements and constraints of the goal?
- Measurable. A measurable goal means that it is possible to identify exactly what it is you will see, hear, and feel when the goal is reached, and no doubt are left. As concrete evidence is needed in order to measure progresses, the goal must be broken down into measurable elements. Defining the physical manifestations of the goal or objective makes it clearer and easier to reach.
- Achievable. The goal should be realistic, based on available resources and existing constraints. For this reason, it is necessary to estimate the effort, time, and other costs the goal will take against the profits and the other obligations and priorities. If the effort is undertaken when enough resources (time, money, talent) are not available, failure is almost certain.
- Relevant. In order to be considered worthwhile, the goal should support or be in alignment with other business objectives. Relevance is the affirmative answer when asking if reaching the goal worth the effort to achieve it.
- Time-bound. The goal must have a deadline or defined end, even if the deadline should always be realistic and flexible in order to keep morale high and avoid a useless and harmful race against time.

Once stories have been written in Pivotal Tracker, and they fit with the principles above, they must be **prioritized** in backlog, dragging them higher or lower in the panel according to their importance for the project development. Once stories are arranged in order of priority, they must be **estimated**. In a planning meeting the entire team discusses each story to gain shared understanding, adds extra information if needed, and collectively estimates each story. Stories

are assigned points by the development team and are estimated by complexity in effort rather than by the time it will take to implement the feature.

Pivotal Tracker allows the estimation of features using a fixed-point scale, chosen at the time of project creation between three default point scales supported: Linear (0, 1, 2, 3), Powers of 2 (0, 1, 2, 4, 8), and Fibonacci (0, 1, 2, 3, 5, 8). Simultaneously, stories can be assigned to team members depending on skills and competences of each one.

At this stage, the concept of iteration comes to attention. An iteration, also called 'sprint', is a period of time, usually one week, used as a unit of measure to calculate expectations according to the work done in each one. As the team prioritizes and estimates stories, Pivotal Tracker divides them into future iterations depending on the team's pace of work in the previous iterations, in order to predict as accurately as possible when future work is going to be completed. Pivotal Tracker offers a transparent team view of priorities, and this makes sure that everyone knows what needs to be done, what is being done, and when it will be completed.

Tracker's agile philosophy not only helps the team in planning work and keeping pace, but also helps in adjusting and changing course when something unexpected happens, so the team can deliver earlier and more consistently.

When stories comes complete with estimation, the button ***Start*** is available on each one. A developer (or one of the developers in charge of processing that story) clicks *Start* on the next story in the current iteration. Now the button available is ***Finish***. The developer collaborate with other team members (e.g. customer, project manager, product owner, designers, testers) to perform coding and testing activities to build the feature increase requested by the story. When coding activities for the story have been completed and they have passed all the automated tests, the developer can click the button *Finish*. Once the CI (continuous integration) build for the newly committed code has passed, the code is deployed to appropriate test environments, and stories are marked as ***Delivered***.

Now the customer, the project manager, or the product owner in collaboration with testers and other team members if needed, verifies that the story has been processed in accordance with the story. Depending on whether the acceptance criteria have been met, the story is ***Accepted*** or ***Rejected***. A rejected story must be newly processed, while an accepted story turns

green and moves to the top of the *current* panel. When the iteration ends, accepted stories move to the ***Done*** panel, divided by their iteration's starting date.

If a delivered feature story is being tested and issues related to that story are found, the most common action is to reject the story and record identified issues in the story's comments. However, sometimes it makes more sense to accept the story, although it presents some issues, and create a separate bug story. It can happen when the issue is minor and not worth delaying a release, or when it is related to multiple stories that would be better manage separately.

Bugs

A bug is a defect in a feature that has already been developed and accepted, and provide no new user value. Workflow of bugs is very similar to that explained above for feature stories: when a defect is detected it is written in Pivotal Tracker, prioritized, and - if needed - assigned.

Bug priority and severity is shown by its position in the *Backlog*: the higher it is, the more important it is to fix.

The developer who takes charge of the bug marks it started when he starts working to fix it. The bug is marked finished when coding and testing activities are completed and deployed to an appropriate test environment. A tester, that can be the customer, project manager, product owner, customer support representative, or another team member, verifies and accepts or rejects the fix.

Focusing on the structure to follow when writing down a bug, it should explain what is the problem detected and what is instead the expected behavior. The steps to reproduce it and all the material useful to make understand it better should be included in the bug description. Bugs do not have points assigned for two reasons: firstly, as said above a bug do not provide new user value, and secondly bugs can be impossible to estimate as it could take few minutes or many days to resolve. Bugs may be tagged with labels relating them to a particular release, feature area, or epic.

Chores

A chore, the third story type, is a story of a technical nature that is necessary but provides no direct, clear value to the user (e.g., “Setup an application server Apache-Tomcat for the application deploy in the staging environment” or “Develop of a batch application for data import and manipulation”).

Chores are not given points, since they do not increase user value directly. This means that if a chore seems to provide user value, then it should be incorporated into a feature story rather than separated out as a chore.

The structure of a chore must be simple and clear in explaining what is needed and how, and it should include instructions and other assets that serve to execute the chore.

Releases

A release is a milestone marker in the backlog that allow the team to keep track of progresses toward concrete goals such as a software launch, a demo, an internal beta or a product release to customers. It is possible to specify a target date for each release, and all stories necessary to accomplish that release should go above the marker for it. Releases only have two states: unstarted and accepted.

4.4.2 Using velocity

As anticipated, Pivotal Tracker plans future iterations in the *Backlog* panel, allocating a reasonable amount of work according to the project team’s *velocity*, which is the team’s pace of work in the previous iterations. Tracker calculates ongoing velocity depending on the average number of story points completed in the past one to four iterations, this value is set when choosing the velocity strategy in Project Settings. In Pivotal Tracker, velocity is the governing force behind every project as it uses it to elaborate predictions on the number of future iterations a given backlog of stories will take to complete, and it is constantly changing based on the actual rate of completion.

For example, if a project's velocity strategy is set to three past iterations whose totaled 8 points, 15 points and 10 points, Pivotal Tracker computes an average velocity of 11, and plans the stories in the backlog for current and future iterations to maintain that average velocity.

It is possible that in the current iteration Tracker planned less points than the average, in our example 10 points instead of 11. The reason is that if the next story in the backlog is a 3-point story, for example, Tracker does not include it because the total would be too high over the average. So the decision to process or not that story is left to the team: when they finish the 10 points they will be able to start the next one. In the case in point, to maintain the average stable over time, Pivotal Tracker plans 12 points for the next iteration.

Project's average velocity is visible at the top of the sidebar and *Current* panel.

Initial velocity

At the beginning of a project, Pivotal Tracker clearly cannot give an average velocity, so for the first iteration it will use the initial value for velocity set in *Project Settings*. It will revert velocity to this default number also when a number of iterations (set for Velocity Strategy, in *Project Settings*) ended totalizing 0 points. As soon as an iteration is completed, Tracker will use the calculated average velocity of the project and that will be the displayed value.

Adjusting velocity

Pivotal Tracker gives the opportunity to experiment hypothetical scenarios overriding calculated velocity of the project with a speculative value, in order to see what the impact on future iterations and releases would be. This experiment is not stored, so other users do not see it and the velocity reverts to its real value clicking the button *Revert* or reopening the project.

During a project it can happen that conditions of the team change. Adjusting team strength and iteration length as needed can ensure velocity is an accurate reflection of the team's workload.

The possibility to modify team strength on the iteration header allows to keep track of any possible variation in the team for each iteration. This helps to take into account temporary team fluctuations such as holidays, sicknesses, or other.

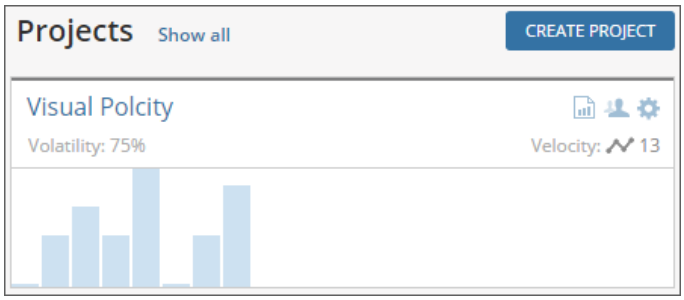
For example, if two of a ten-people team are on holiday, the team strength should be set at 80% for that iteration. In the same way, if you put in three new resources your team strength should be set at 130%. When an iteration's team strength is customized, the points done in that iteration are converted into the number of points the team would have done as if team strength was 100%. Let's suppose our example team totalize 20 points in an iteration: in the first case (TS 80%) the number of points actually counted is 25, while in the second case (TS 130%) the number of points will be 15.

It can happen that an iteration presents a 0% team strength, in this case that iteration will not be taken into account from velocity calculation.

4.4.3 Volatility

Another useful tool in Pivotal Tracker is *volatility*. It is a percentage reflecting the project velocity variation from iteration to iteration, which shows the consistency of the team work output. It is calculated on the basis of the number of recent iterations set for project's velocity strategy in project's settings. This value is shown on the dashboard, in the project box (Figure 14).

Figure 14 - Volatility shown of the project box



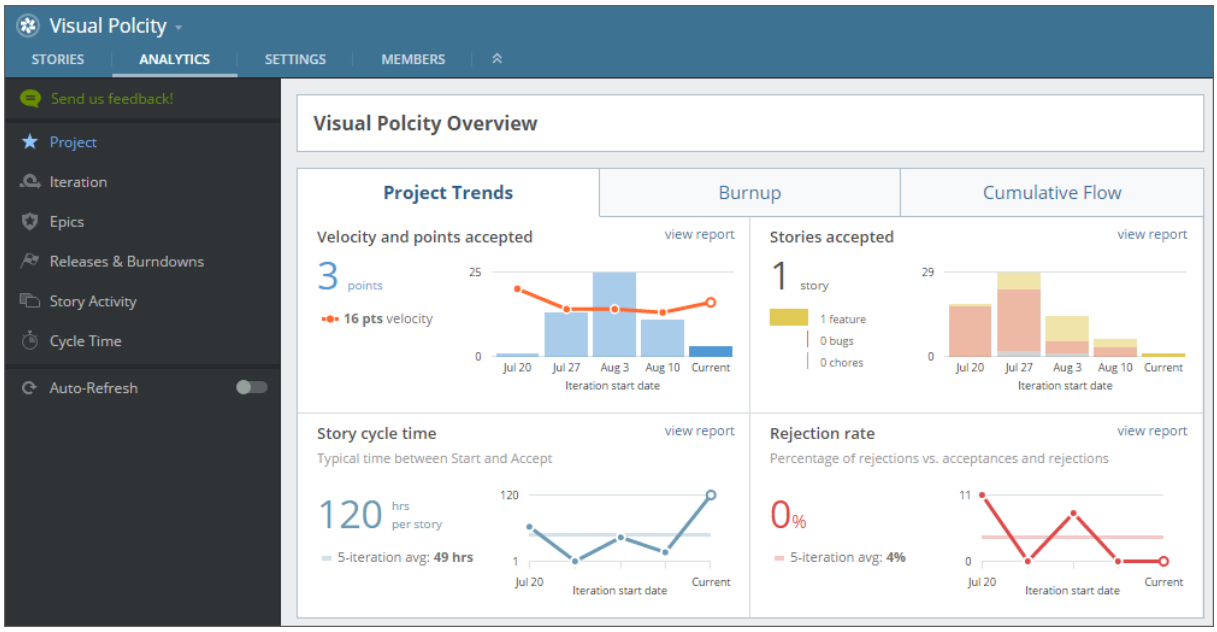
Source: pivotaltracker.com

As seen above, velocity is an average that can vary significantly from iteration to iteration for many reasons, for example it decreases when the team starts using a new technology that slow developers down as they learn, or it increases when business priorities shift and new resources are put into the team. These variations in project velocity between iterations cause a high-value volatility and although that means a low consistency in team work output it is not necessarily a negative thing. An analysis of the reasons of a high volatility should be done anyway in order to understand if it is necessary to take measures together with the team to pave the way to a consistent pace.

4.4.4 Analytics

Analytics is a section of the project management interface where are charts and reports giving an overall project picture and useful trends in the project (Figure 15).

Figure 15 – Project Trends giving an overall picture of the project



Source: pivotaltracker.com

Analytics are designed to give a picture of what is being processed, how work is going on, and where the team is lacking and might need help. They show the general progress on work in order to help in identifying problems or bottlenecks, and also give the chance to analyze in detail specific stories, iterations and epics.

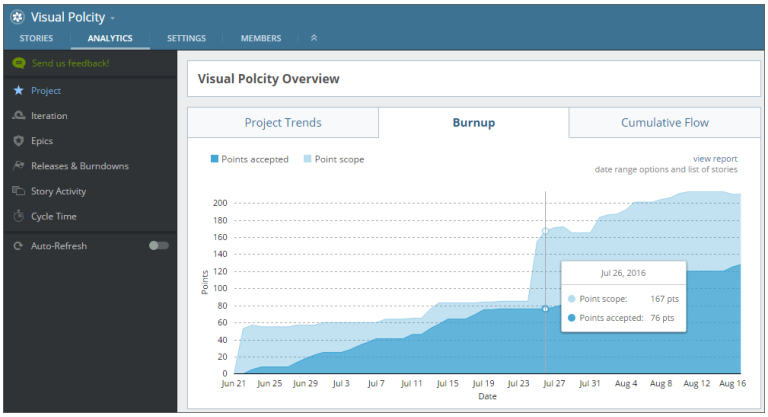
Project Overview show high-level metric data on the project and allows instant access to other charts and reports useful for best managing the project. Moving on velocity, which has already been explained, other important charts are briefly discussed below.

Story Composition, in the second box, shows the distribution of each type of story in iterations.

Cycle Time, also reachable from the side menu, give an overall view of what happens to stories from different points of view: it shows the time spent on stories by state, points, type, or a median of all, and displays all the stories in a chosen iteration with information on cycle time for each. Part of the Cycle Time report is also a graph showing the **rejection rate**, displayed also

in the fourth box, which shows the rate at which stories were rejected and the time before their restart.

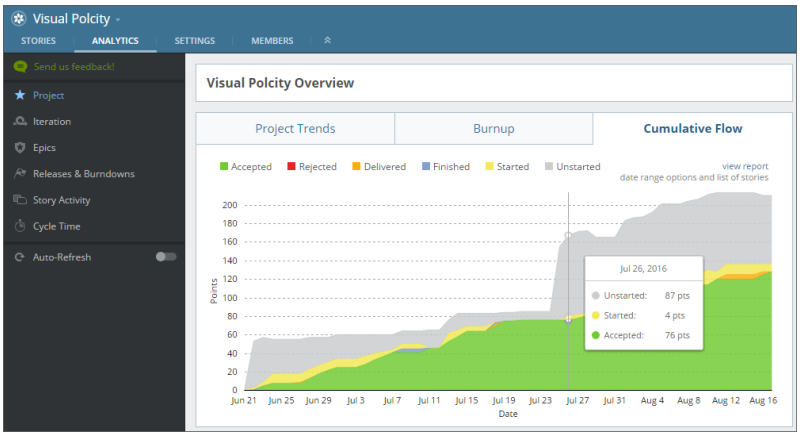
Figure 16 - Burnup chart



Source: pivotaltracker.com

Burnup (Figure 16) shows how many points have been accepted out of the total amount of points for the chosen iterations, in order to track the project work progress toward its completion.

Figure 17 - Cumulative Flow chart



Source: pivotaltracker.com

Cumulative Flow chart (Figure 17) displays for each day over time the exact number of stories in each state: accepted, rejected, unstarted, started, finished, and delivered.

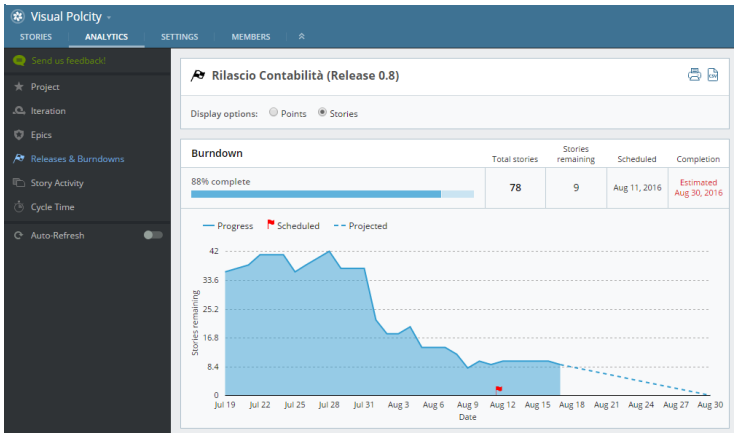
Iteration report gives an overview of expected progress, which is the acceptance of stories needed day by day in a given iteration in order to reach the goal.

Epics report gives an image of the progress of epics and large themes, with the chance to go in depth of each one and understand how the work area has changed over time. For each epic the table shows the progress of stories or points, depending on which option has been chosen, in term of the "situation" they are in: total (point or stories), accepted, in progress, unstarted, iceboxed, date of completion if present.

Releases and Burndowns report is almost the same as Epic report but instead of epics, a list of completed, in-progress, and upcoming releases are displayed. Each one has a bar showing its progress in terms of points or stories, and a link named 'view burndown'. If clicked, the link

brings to a **burndown chart** showing the number of points (or stories) remaining to reach the target for that release (Figure 18). The burndown graph helps in tracking progresses and in identifying, with the burn-down dotted line, the trend the team should keep to reach the target on the scheduled deadline assuming the current velocity is kept stable. This chart gives further useful information that can be seen at a glance, such as the moments along the releases when stories has been added and consequently the project scope increased.

Figure 18 - Burndown chart



Source: pivotaltracker.com

Story Activity report shows all the activities done for each story worked day by day in a range of time that can be chosen between an iteration or a defined period of time, also selecting dates manually.

Pivotal Tracker is an essential strategy tool for managing a software development project with Agile, because it allows to put together many different boundary objects in a single tool. Indeed, in every user story it is possible to attach files containing analysis documents, prototypes, photoshops, and so on. But here is why Tracker is so special. As seen above, on one side there are estimates on how difficult tasks to complete stories could be for the team, and on the other side there is the time the team actually takes to complete these stories. Pivotal Tracker puts together these two elements and uses their correlation to elaborate estimates on how long the project workload will take to conclusion.

All the graphs and charts provided by Pivotal Tracker make it clear to understand how the project is going, if corrective measures should be taken and how, and it is an optimal tool to facilitate communication about features and bugs issues between the team and the clients.

When using Pivotal Tracker it is easy to realize how it helps in organizing the team. Whether deadline are self-imposed, or is the client to impose deadlines depending on his necessities, it

is clear how Tracker helps in understanding which issues are priorities and which can be put aside and processed after the deadline.

4.5 Conclusions

As previously seen, boundary objects represent an essential tool for collaboration between development team members and product stakeholders, since they support organizations and people in overcoming difficulties emerging from their different backgrounds. In particular, boundary objects help in building a common language, shared understanding, processes integration, goals convergence, and mutual transparency, everything aimed to achieve the common goal of developing a software in compliance with product owner's requirements and user's needs as efficiently as possible. Boundary objects and strategy tools represents also some challenges emerging from their flexibility, which is an advantage but also a pitfall if not managed properly, as outlined above. For this reason a deep knowledge of the project, team, goals, and tools to use, is crucial for an effective management of a software development project with strategy tools like Pivotal Tracker.

CHAPTER 5. CONCLUSIONS

Recalling what was outlined in the Introduction, project-based organizations are organizational forms which are optimized to use temporary systems in order to achieve specific work targets. In other words, organizations whose primary focus is to manage and carry out a specific project, from beginning to end. The increased number of project-based organizations shows the current trend of an extremely competitive, and constantly evolving global economy, which demands consistent business results to survive.

Since project-based organizations are entirely dedicated to one or more projects, competitive advantage can only be gained with a strategy that fully aligns with, and encompasses the whole organization. The organization must be aimed to gain better results, and deliver business value. The crucial role played here is by a strong commitment to optimizing project management, and it is the reason why this dissertation is concentrated on the evolution of methodologies and the different types of project scope.

The purpose of project management is to focus on the following key components of a project: ensure quality, manage changes and problems, integrate with the organization's business processes and systems, handle project knowledge, and ensure business innovation.

In the early 1900s, the project management discipline began to develop in response to the need to have high control over very complex and risky projects. Since that time, it has continued to evolve. Traditional project management is characterized by a rigid structure, in which the whole plan of the project must be outlined before its start. It is inflexible, which makes it unsuitable to adapt to any changes during the project's development. The planning is rigorous and accurate, and it must be followed in steps, for example like a waterfall. The result is discovered only in the end, since controlling or quality assessment activities during the venture are not done.

The first changes in an organization management occurred when companies began to feel an increasing pressure from the market, because of an increase in national and international competition, and increasingly demanding customers. In order to gain competitive advantage, leaders and CEOs began to understand that they needed more flexibility, and increased speed and effectiveness while managing their projects.

The first radical change occurred during the 80s with the introduction of Stage-Gate, and concerned the physical product development field, as it was still the major industry. The Stage-Gate process emerged in response to the pitfalls of traditional project management, which represented a challenge when facing new product development process. Stage-Gate ensures a high degree of regulation and control. This is thanks to its well-defined framework that is structured in stages, where work activities are carried out under a standardized planning and scheduling, and gates, where designated members decide whether to continue to invest in the project, or interrupt it on the basis of regulated procedures aimed to assess its quality.

Stage-Gate had a lot of success, but the primary reason it continued to evolve was because of the need companies had to overcome the sequential design and the pitfalls of the rigid, traditional system. What emerged was an optimized and increasingly scalable and flexible process that could meet the needs of many different project types, therefore allowing companies to fit the model to their own specific organizational requirements. The Stage-Gate process is still widely implemented and appreciated by manufacturers all over the world thanks to its ability to drastically improve new product development processes.

Shortly after the rise of Stage-Gate, the software development industry began to need a new project management method. Stage-Gate did not fit well with software development, as it was a methodology created for physical products, and implemented in a sequential way. Over the years, programmers found that it was not feasible to develop a software following stages rigidly, because they commonly found themselves having to go back and change something, before moving forward with the software. This is due mainly to their clients' uncertainty about the final product's features; it is possible that a customer asking for the development of a new physical product does not know exactly what he wants to obtain in the end. This is even more common with software development clients, since the product is an abstract entity at its beginning. It is evident here that traditional project management, because of its rigidity, is an impractical strategy for software development project management.

Agile project management was created in response to the needs for a flexible product management structure for the software development industry. It overcame the issues of traditional project management, which have been discussed above. In particular, Agile introduced revolutionary concepts outlined in the Agile Manifesto and Principles; a document

that helps development teams determine the practices that are most important for a successful project. It encourages communication, collaboration, adaptation, and the evolution of the project itself, through its main characteristics of frequent deliveries, flexibility, and continuous feedback from stakeholders.

Many new development methods arose after Agile's success. Each one differed from the others by some particular characteristic: simplicity and dynamism for Extreme Programming, attention to the project's criticality and complexity for Crystal, prioritizing a fixed time and budget over number of functionalities developed for Dynamic Systems Development Method, and continuous improvement for Kanban. Scrum deserved a more in-depth analysis since it is now the leading Agile development methodology, used by the top companies around the world. With its structure based on time-boxed sprints, Scrum provides a framework that encourages communication and collaboration, flexibility to project changes throughout the whole development, motivation and commitment of the entire project team, continuous feedback, transparency, and an iterative development planning. Benefits of Scrum can be summarized into these main outcomes: increased customer satisfaction, improvement of team morale, increased transparency, increased quality corresponding to decreased risk, and higher speed and efficiency.

Therefore, Scrum was born originally as a methodology for new product development, later becoming a method for managing software development projects, and returning in recent times to represent a management method applied to other types of projects also. Aware of the advantages accessible by using Agile, many manufacturers using Stage-Gate for their product development projects began to integrate and blend the two methods, giving life to Agile-Stage-Gate hybrids. Some adaptations of Agile have been made in order to fit to the physical product industry, but it has largely been demonstrated that hybrids are bringing many positive outcomes to these far-sighted companies. Agile-Stage-Gate hybrids, even though presenting some challenges, blend the best sides of both methods, which generates higher efficiency, speed, focus, and more productive frameworks for coordination and communication.

The focus on agile contracts aimed to show that when managing a project with agile methodologies, it is crucial to draw up appropriate contracts that are specifically drafted for the project they have to regulate. This makes them able to follow the values and characteristics of the agile approach while being in line with the project's goals. Before drafting an agile

contract, it is crucial to deeply understand agile approach principles, the flexible type of development it implies, and consequently its impact on the entire business relationship regulation between parties.

Many possible contract models exist, making many adaptations possible for customers and suppliers who want to find the best way to manage their agile project. The factors to keep under consideration, and to work on carefully when writing an agile contract, have been explained to help the reader understand how they are complexly related and that they can give life to many different contract models. However, in any way that a contract is framed, trust, collaboration, and transparency, are key aspects that are more likely to lead to a successful and profitable project for both parties.

The chapter on the company case study helped this dissertation to focus on boundary objects, which are all those essential tools that enhance collaboration between development team members and product stakeholders. Since organizations and team members come from many different backgrounds, they need something helping them to overcome the difficulties that emerge because of cultural and social diversities. Boundary objects are physical objects able to adapt to the social worlds' needs while maintaining their structural identity across actors. In particular, boundary objects help in building a common language, shared understanding, goal convergence, processes integration, and mutual transparency, all elements aimed at achieving the goal of the software, in compliance with the product owner's requirements, and the user's needs, while still being as efficient as possible. Organizing all the information and boundary objects on a common ground gave rise to agile boundary objects management tools. These are shared environments that are reachable by everyone involved in the project. All the team members collaborate to organize their work. They are organized repositories of data, information, and boundary objects, which besides all the benefits brought by boundary objects in themselves, offer all the advantages deriving from the organization of all of them, if correctly done. Pivotal Tracker, the management tool analyzed in the company case study, is used to organize user stories. User stories are the most common software development boundary object, and they can be attached with other boundary objects, such as analysis documentation, prototypes, and so on. In addition, thanks to this ability to organize the work, as well as all of the project analytics they provide, these tools support managers in decision making and in crafting organizations' competitive strategies.

Boundary objects and strategy tools represents also some challenges emerging from their flexibility. For example, holding too much information, or trying to connect too many different backgrounds, can become confusing and lead to users abandoning them. Flexibility is an advantage, but it also has its fallbacks if not managed properly. A deep knowledge of the project and its team, goals, and tools, is crucial for the effective management of a product development project with boundary objects, whether they are physical products, or software.

The table below summarize the main points of boundary objects.

Table 2- Main aspects of boundary objects

	Boundary objects in physical products	Boundary objects in software products
Functions	- Provide a common ground for **communication** between team members with different social and cultural backgrounds. - Build a **shared understanding** of information across boundaries. - Improve **process integration**. - Improve mutual **transparency**.	- Facilitate **communication** between team members aimed at the **shared understanding** of project flows, interactions, and functionalities. - Allow **process integration**, otherwise very difficult without boundary objects. - **Ensure** transparency.
Objects	Sketches, User stories, Charts, Design specifications, Physical prototypes.	User stories, Design specifications, Interactive prototypes, Organizational tools.
When used	Design specifications, charts, and sketches are used mainly for the planning phase at the beginning of the project, before the actual development starts. Physical prototypes are produced later. User stories cover the whole of the project period, since they are used to organize the work flow.	In Agile development, all the boundary objects are constantly used during the project lifecycle, given the iterative nature of the method. Even though specifications are set out at the beginning of the project, they are subject to evolution, changes, and improvements.
Pros and Cons	**Pros** - Alignment of project scope and goals. - Adaptability to different needs while maintaining their identity. - Means of communication recognizable by several social worlds, which act as temporary bridges or anchors. - Standardize methods, which leads to coherence. - Increase commitment of team members. - Increase project quality. **Cons** - Spending too much time on producing documentation often results in heavy and complex material that is difficult to manage and understand by team members: giving detailed directions must be supported by a compromise between level of detail and time spent to produce them. - The intersection of too many different social worlds can cause confusion and mismatches, because of the many different meanings the object is given. This undermines the object's functions.	**Pros** - Significant enhancement of communication and collaboration minimizing the risk of understanding mismatches. - High involvement and commitment of all the team members, stakeholders included. - Speed up development processes. - Facilitate reviewing and testing process. - Increase project quality. **Cons** - In cross-cultural teams, boundary objects may accentuate the status differences of culturally diverse groups, which may subsequently lead to conflict and negative stereotyping between groups. - Too much flexibility can result in a chaotic work environment and mismatches in understanding, causing a decrease in team performance. - Objects containing too much information or trying to connect too many background can become confusing, which may lead to their abandonment.

Source: elaboration of data collected from sources mentioned in paragraph 4.3.1.

In conclusion, this dissertation assessed the evolution of project management, analyzing the main methods that emerged over time according to the changing needs and nature of projects. The importance of project management has been highlighted throughout this work, from traditional to modern methods, focusing on two industries in particular: physical products, and software. It has demonstrated how the original methodologies born to fulfill the needs of a particular field, gave life to hybrids, which bring about their own advantages. The component that has been consistent throughout this paper have been boundary objects. They have been critical in the many methodologies outlined, the evolution of projects, and across all industries. Boundary objects are all the elements used in a project that enhance collaboration and shared understanding, ensure transparency, improve commitment, speed up the project lifecycle, guarantee process integration, and involvement of team members, and bring all the derived benefits, including quality, provided that they are appropriately used. It can be concluded that boundary objects are an essential element both in new product development, and in software development projects, as they successfully help in delivering a product consistent with project requirements.

BIBLIOGRAPHY

Apke L., (2015). “Understanding the Agile Manifesto. A brief & bold guide to Agile”, Lulu Publishing.

Arbogast T., Larman C., Vodde B., (2012), “Agile Contracts Primer”, Agile Contracts.

Awad M.A., (2005). “A comparison between Agile and Traditional Software Development Technologies”, University of Western Australia.

Barrett M., Oborn E., (2010). “Boundary Object Use in Cross-Cultural Software Development Teams”, Human Relations, 63(8) 1199-1221.

Bayer H., Holzblatt K., Baker L., (2004). “An agile user-centered method: Rapid Contextual Design”. Extreme Programming and Agile methods – XP/Agile Universe 2004, Lecture notes in Computer Science, 3134, 50-59, Springer.

Beck K., (2005). “Extreme Programming Explained: Embrace Change”, Addison-Wesley.

Bechky B.A., (2003). “Workplace Artifacts as Representations of Occupational Jurisdiction”, The American Journal of Sociology, 109 (3) 720-752.

Begel A., Nagappan N., (2007). “Usage and perceptions of agile software development in an industrial context: An exploratory study” in *First International Symposium on Empirical Software Engineering and Metrics*, IEEE Computer Society.

Boehm B., Turner R., (2004). “Balancing Agility and Discipline: A Guide for the Perplexed”, Addison Wesley.

Brotherton S.A., Fried R.T., Norman E.S., (2008). “Applying the Work Breakdown Structure to the Project Management Lifecycle”, PMI Global Congress Proceedings, Denver, Colorado.

Carroll J., (2012). “Agile Project Management for speedy results”, EasySteps Limited.

Cattani G., Ferriani S., Frederiksen L., Täube F., (2011). “Project-Based Organizing and Strategic Management”, Emerald Group Publishing Limited.

Cheng L., Subrahmanian E., Westerberg A.W., (2003). “Design and planning under uncertainty: issues on problem formulation and solution”, Computers and Chemical Engineering 27 (6), 781-801.

Clark D.N., (1997). “Strategic Management Tool Usage: A Comparative Study”, Strategic Change 6(7): 417–27.

Cockburn A., (2007). “Agile Software Development: Software development as a Co-operative game”, Addison-Wesley.

Coffin R., Lane D., (2006). “A Practical Guide to Seven Agile Methodologies”, Part 1, XP, Scrum, Lean, and FDD.

Collier K., (2012). “Agile Analytics: A Value-driven Approach to Business Intelligence and Data Warehousing”, Addison-Wesley.

Cooper R.G., (1990). "Stage-gate systems: A new tool for managing new products", Business Horizons.

Cooper R.G., Edgett S.J., (2005). “Lean, Rapid, and Profitable New Product Development”, Product Development Institute.

Cooper R.G., (2008). “Perspective: The Stage-Gate Idea-to-Launch Process – Update, What’s New and NexGen Systems”, J. Product Innovation Management.

Cooper R.G., (2009). “How companies are reinventing their idea-to-launch methodologies”, J. Product Innovation Management.

Cooper R.G., (2014). “What’s next? After Stage-Gate”, Research Technology Management 57(1).

Cooper R., (2016). “Agile-Stage-Gate Hybrids”, Research-Technology Management 59(1).

Dönmez D., Grote G., Brusoni S., (2016). “Routine interdependencies as a source of stability and flexibility. A study of agile software development teams”, Information and Organization 26(3), 63-83.

Dudziak T., (2000). “eXtreme Programming, an overview”, Methoden und Werkzeuge der Softwareproduktion WS.

Fowler M., Highsmith J., (2001). “The Agile Manifesto”, Software Development, 9(8), 28-35.

Golany B., Shtub A., (2001). “Handbook of Industrial Engineering: Technology and Operations Management”, John Wiley & Sons.

Highsmith J.A., (2002). “Agile software development ecosystems”, Addison-Wesley Professional.

Hinds T., (2014). “The four values of Agile Manifesto”, Neotys Blog.

Hobday M., (2000). “The project-based organisation: an ideal form for managing complex products and systems?”, Brighton, UK: Research Policy 29.

Holitza M., Ambler S.W., (2012). “Agile for Dummies”, John Wiley and Sons.

Jurca G., Hellmann T.D., Maurer F., (2015). “Integrating Agile and User-Centered Design - A Systematic Mapping and Review of Evaluation and Validation Studies of Agile-UX”, Department of Computer Science, University of Calgary, Canada.

Karlström D., Runeson P., (2006). “Integrating Agile software development into Stage-Gate managed product development”, Empirical Software Engineering (11), 203-225.

Kerzner H.R., (2013). “Project Management: A Systems Approach to Planning, Scheduling, and Controlling”, Wiley.

Kodama M., (2007). “Project-Based Organization in the Knowledge-Based Society”, London, Uk: Imperial College Press.

Lacey M., (2012). “The Scrum field guide”, Addison-Wesley Publishing.

Larman C., (2004). “Agile and Iterative Development: A Manager's Guide”, Addison-Wesley.

Larman C., Vodde B., (2010). “Practices for Scaling Lean & Agile Development: Large, Multisite, & Offshore Product Development with Large-Scale Scrum”, Addison Wesley.

Lawson D., (2009). "PMBOK Quick Implementation Guide”, Emereo Publishing.

Layton M.C., (2015). “Scrum for Dummies”, John Wiley and Sons.

Leithold N., Haase H., Lautenschläger A., (2015). “Stage-Gate for SMEs: a qualitative study in Germany”, European Journal of Innovation Management.

Mall R., (2004). “Fundamentals of Software Engineering”, Prentice Hall of India.

Martin R.C., (2004). “Estimating Costs Up Front”, Extreme Programming mailing list.

Martin R.C., Martin M., (2006). “Agile Principles, Patterns and Practices in C#”, Prentice Hall.

Maximini D., (2015). “The Scrum Culture. Introducing Agile methods in Organizations”, Springer International Publishing.

Mnkandla E., Dwolatzky B., (2004). “A survey of Agile Development Methodologies”, the transactions of the SA institute of electrical engineers: 236-247.

Morris R. A., McWhorter S. B., (2008). “Project Management That Works: Real-World Advice on Communicating, Problem-Solving, and Everything Else You Need to Know to Get the Job Done”, Amacom Books.

Nonaka I., Tackeuchi H., (1995). “The Knowledge-Creating Company: How Japanese Create the Dynamics of Innovation”, Oxford University Press.

Norman E. S., Brotherton S. A., and Fried R. T., (2011). “Work Breakdown Structures: The Foundation for Project Management Excellence”, Wiley.

Ohst D., Welle M., Kelter U., (2003). “Difference tools for analysis and design documents”, International Conference on Software Maintenance, IEEE.

Opelt A., Gloger B., Pfarl W., Mittermayr R., (2013). “Agile Contracts. Creating and Managing Successful Projects with Scrum”, Wiley.

Persson J., (2014). “Communication through boundary objects in distributed agile teams”, Linkopings Universiteit.

Pichler R., (2010). “Agile Product Management with Scrum”, Addison-Wesley Professional.

Poppendieck M., (2005). “Agile contracts”, Agile Conference Workshop.

Product Development Institute, (2016). “Stage-Gate – Your roadmap for New Product Development”.

Project Management Institute, (2013). “A guide to the Project Management Body of Knowledge – Fifth Edition (PMBOK Guide)”. Newtown Square, Pennsylvania: Project Management Institute.

Rad N.K., Turley F., (2013). “The Scrum Master training manual”, Management Plaza.

Reagan B., (2012). “Going Agile: CA Technologies, Clarity PPM Division’s transformative journey”, Digital Celerity.

Rossberg J., (2014). “Beginning Application Lifecycle Management”, Apress.

Scanlon Thomas E., (2011). “Breaking the addiction to process - An introduction to agile project management”. IT Governance Publishing.

Schwaber K., (2004). “Agile Project Management with Scrum”, Microsoft Press.

Schwaber K., Sutherland J., (2016). “The Scrum Guide. The definitive Guide to Scrum: the rules of the game”, Scrum Guides.

Sommer A.F., Hedegaard C., Dukovska-Popovska I., Steger-Jensen K., (2015). “Improved product development performance through Agile/Stage-Gate hybrids: The next-generation Stage-Gate process?”, Research-Technology Management 58(1).

Spee A.P., Jarzabkowski P., (2009). “Strategy tools as boundary objects”, Strategic Organization 7(2), 223-232, Sage Publications.

Star S.L., Griesemer J.R., (1989). “Institutional Ecology, `Translations’ and Boundary Objects: Amateurs and Professionals in Berkeley's Museum of Vertebrate Zoology, 1907-39”, Social Studies of Science, 19(3), 387–420.

Stellman A., Greene J., (2015). “Learning Agile: understanding Scrum, XP, Lean and Kanban”, O'Reilly Media.

Sutherland J., (2014). “The Art of doing twice the work in half the time”, Crown Business.

Sydow R., Lindkvist L., DeFillippi R., (2004). “Project-Based Organizations, Embeddedness and Repositories of Knowledge: Editorial”, SAGE publications.

Takeuchi H., Nonaka I., (1986). "New Product Development Game", Harvard Business Review.

Taylor J. C., Lashman R., Helling P., (1994). “Practical Problem-Solving Skills in the Workplace”, Amacom.

Thiry M., (2006). “Beyond the Matrix: The Integrated Project-Based Organization”, PMI Global Congress Proceedings.

Van Amerongen R., (2008). “Agile software development, the principles”, Amis Technology Blog.

Voigt B., (2004). “Dynamic System Development Method”, University of Zurich.

Wagener J., Schmit S., Mandal A., Rajendran V., (2012). “Project Management Using Kanban”, University of Luxemburg.

Weiss J. W., Wysocki R. K., (1992). “Five-Phase Project Management: A Practical Planning and Implementation Guide”, Perseus Books.

Westcott R. T., (2004). “Simplified Project Management for the Quality Professional: How to Plan for and Manage Small and Medium-Size Projects”, ASQ Quality Press.

Whitaker K., (2009). “Principles of Software Development Leadership: Applying Project Management Principles to Agile Software Development”, Cengage Learning.

Williams C., (2013). “Agile Manifesto: working software over comprehensive documentation”, Source Allies Blog.

Wrubel E., Gross J., (2015). “Contracting for Agile Software Development in the Department of Defense: An Introduction”, Software Engineering Institute, Carnegie Mellon University.

Wysocki R., (2013). “Effective Project Management: Traditional, Agile, Extreme”, Wiley.

Zaitsev A., Gal U., Tan B., (2014). “Boundary Objects and Change in Agile Projects”, Australasian Conference on Information Systems, ACIS.