The use of methodologies for the development of IT projects

Lorenzo Cimasoni
Rue François-Guillimann 5
1700 Fribourg
lorenzo.cimasoni@unifr.ch

Supervisor
Dr. Stefan Hüsemann

November 2009
Abstract

Nowadays there are dozens of methodologies which could be chosen in order to manage Information Technologies projects. The evolution of those methodologies is due mainly to the objective of having a suitable and functional method that could help the development of different projects, reducing at the minimum the possibility of failure.

The aim of this work is to comprehend and give a theoretical and practical perspective of the different methodologies which are currently used to manage IT projects.

The topics that will be discussed are: the work of project management, the various existing methodologies and the differences between them, how tailoring them, and finally some case studies based on real projects.
I want to thank...

Dr. Stefan Hüsemann who followed and helped me throughout this work,
the IT development team of the SiSO SA for their kind availability,
my parents, Wendy, and Tanja for their constant help and support.
Contents

1. Introduction ................................. 1
   1.1. Problem description ......................... 1
   1.2. Questions and subjects to be treated .......... 2
   1.3. Methodology and Structure of the Thesis ....... 3

2. Introduction to project management ............. 4
   2.1. Introduction ................................ 4
   2.2. Project management ......................... 4
   2.3. Information Technology projects .......... 6
   2.4. Success and failure ....................... 7

3. Introduction to methodologies .................. 13
   3.1. Introduction ................................ 13
       3.1.1. Standards ............................... 16
   3.2. Waterfall .................................. 17
   3.3. Spiral ...................................... 19
   3.4. IBM Rational Unified Process .............. 21
   3.5. HERMES ..................................... 24
       3.5.1. HERMES and RUP ......................... 25
   3.6. Agile methods .............................. 26
       3.6.1. Scrum ................................... 29
       3.6.2. eXtreme Programming .................... 32
       3.6.3. OpenUP .................................. 35

4. Tailoring with methodologies ................... 38
   4.1. Introduction .............................. 38
       4.1.1. Unified Method Architecture .............. 39
   4.2. Rational Method Composer ................... 40
   4.3. Eclipse Process Framework ................... 43
List of Figures

2.1. The Project Management Triangle ........................................... 8
2.2. Graphic of standish group statistics ......................................... 9
2.3. Traditional and XP assumption about the cost of change over time .... 10
2.4. Problem resolution and fault-finding technique .......................... 11

3.1. The waterfall model .............................................................. 18
3.2. The spiral model ................................................................. 19
3.3. RUP overall architecture ....................................................... 21
3.4. RUP’s iterative model .......................................................... 23
3.5. HERMES phases diagram with possible iterations ....................... 25
3.6. HERMES metamodel ............................................................ 26
3.7. Plan-driven versus value-driven methods .................................. 27
3.8. The Scrum process model ...................................................... 29
3.9. An example of a Scrum sprint burn down chart ........................... 32
3.10. eXtreme Programming lifecycle ............................................ 33
3.11. eXtreme Programming user story example ............................... 35
3.12. OpenUP iteration lifecycle ................................................... 36
3.13. OpenUP lifecycle model ....................................................... 37

4.1. UMA concept of method content and process separation ............... 39
4.2. RMC method content, task example ....................................... 42
4.3. EPF Ecosystem ................................................................. 44

5.1. SiSO platform - Planning (a), Construction (b) and Operation (c) phases 50
5.2. SiSO projects organisation ..................................................... 51
5.3. SiSO test development platform ............................................. 54
5.4. e-dec architecture ............................................................. 56
5.5. all e-dec projects organisation ............................................. 57
5.6. RUP artifacts used in e-dec project ....................................... 59
5.7. e-dec planning approach ........................................... 61
5.8. Representation of the SiSO SA Projects lifecycle .................. 63

B.1. EPF web pages - Artifact: Sprint Burndown Chart ................ 75
B.2. EPF Composer authoring tool ........................................ 76
List of Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1</td>
<td>Similarities between OpenUP and Agile Manifesto</td>
<td>36</td>
</tr>
<tr>
<td>5.1</td>
<td>SiSO modules</td>
<td>48</td>
</tr>
</tbody>
</table>
1. Introduction

1.1. Problem description

All types of projects have some characteristics in common, among which: an objective to be achieved, the uniqueness of the project, the temporary endeavour, and the availability of limited resources.

In order to manage a project is necessary to apply some techniques of project management with the purpose of achieving the goals, especially in the case of increased projects size and complexity.

The thematic of project management is examined in detail in the literature, in the theoretical and technical point of view, including the use of methodologies for the development of Information Technologies (IT) solutions.

Nowadays there are dozens of methodologies, called also software development models, which could be chosen to manage IT projects. The evolution of those methodologies is due mainly to the objective of having a suitable and functional method that could help the development of different kind of projects, thus reducing at the minimum the possibility of failure. This is also due to the fact that there has been an exponential growth of importance and investments on IT solutions in the last decades which has touched almost all aspects of the daily life.

Currently there is no perfect ways to execute a project but a set of many different methodologies where the project manager could be chosen among the most suitable for a specific situation. For example there are methodologies which focus more on the management side of the project, and others that are more concerned with the engineering side. Some covers the entire lifecycle of a project, others tend to guide only the development side.

The choice of the methodology could depend on many factors from personal preferences of the project manager to the type of project. Many methods can also be easily adapted to the project through a work of tailoring which allows to use only the needed parts. This factor can give more flexibility allowing the use of several methods simulta-
1.2 Questions and subjects to be treated

In order to treat all subjects related to IT projects methodologies a series of questions, listed here below, will be discussed through this work:

- What is a (IT) project? This question will be dealt with in the introduction part, with the existing literature, to provide an introduction and allow the further development of the work. [Chapter 2]

- Which are the factors of success for a project? This question is related to the definition of IT project that will be mainly developed in the introduction part of the work. Above all this question intend to introduce the factors of success and failure in IT projects by involving the statistics and the literature. [Chapter 2.4]

- What is a methodology? This question will be treated following the existing literature in order to present an initial overview on the subject, resuming some existing and most used methodologies for project management. Some of these methodologies will be chosen and described more in details permitting to illustrate the general evolution in this sector from waterfall model to more recent agile methodologies. [Chapter 3]

- What are the existing methodologies? How often are they used in IT projects? This question will introduce some different methodologies used in the world of IT projects and try to reflect how much they are spread and used. The importance of this question is to pin down an image of the different methods by extending the different descriptions. [Chapter 3]

- Which are the common points and differences between selected methodologies? This question analyses the different methods in order to find both common points and main differences between them. Its goal is to introduce also the concept of the Unified Method Architecture and deepen it in relation to methodologies. [Chapter 3]

- How to do "tailoring" of methodologies? This question discusses the various utility that allows the tailoring within a particular methodology, in practise it takes only useful pieces for a specific project. This question is important in order to introduce
topics such as the Rational Method Composer and the Eclipse Process Framework. [Chapter 4]

• Which are the positive and negative experiences of IT projects that use methodologies? This question will be developed mainly from interviews made for the preparation of case studies. The importance of this question is to show a more concrete vision of methodologies through practical examples and experience of using them in IT projects. [Chapter 5]

1.3. Methodology and Structure of the Thesis

The aim of this work is to comprehend and give a perspective of the different methodologies that are used nowadays to manage IT projects.

In order to give a wider perspective on the use of those project methodologies, this work will be subdivided into two parts.

The first part will be more theoretical and based on the existent literature. This part will begin with an introduction on the project management, afterwards some IT project methodologies will be presented to give a perspective of the evolution and the actual state of art in this sector, and at the end some possible applications of tailoring on the methodologies will be shown.

The second part concerns a more practical view on the subject with the introduction of two case studies from real IT projects.

These cases studies resulting from the interview with some project managers will be presented with the aim to pin down a more concrete vision of the use of methodologies within specific projects or company situations.
2. Introduction to project management

2.1. Introduction

This chapter presents the theme of project management. It will answer two research questions presented in the introduction:

- What is a (IT) project?
- What are the factors of success for a project?

The first part will introduce the general concept of project and the subject of project management, subsequently will be listed all the general characteristics that differentiate a project IT from the other kinds of project. At the end will be discussed the concept of respect of the constraints in a project as well as the collateral factors, the statistics of success, and the failure in IT projects.

2.2. Project management

The first thing to be done is to define what a project is. A project, as explained in [Cha03, p.22], could be described as a temporary work, a one-time event that has some fix points amongst which:

- A start and an end date.
- Schedule, cost, and quality constraints.
- Is a unique endeavour and contains risks.
- A certain scope that needs to occur.
The end of a project could be understood as reached when the project’s objectives are achieved. In certain this end could be decided when the objectives will not or cannot be achieved, or when the need for the project no longer exists. The result, which may be a product or service, is unique and distinguishable from all other results.

The temporary aspect of a project does not necessarily apply to the result of the project, and the product or service often outlive the project. This definition helps to distinguish what is defined as ordinary maintenance, that follows in many cases the project, to the project itself. Projects are also differentiated from operations which are ongoing and repetitive and have regular result.

Generally there are more others key characteristics that may help describe the term project, some of which are: [BH99, p.2]

- a required plan
- involving into the work several specialists
- the size and the complexity
- non-routine tasks
- several phases in which the work is carried out

Afterwards one of the characteristics of a project described in the list above is that it is usually divided into sequential phases, also known as stages. All this phases gathered together to form what is known as project lifecycle.

There are a lot of examples of what a project is, from building a tunnel, to write a new computer operating system, or write a bachelor thesis. The projects can be developed in different fields from academy research to civil engineering.

Another aspect of the projects is all individuals or organisations involved which directly or indirectly have an influence on it. These individuals are called stakeholders, and they can cover different roles inside a project, from upper level management, team members, customers to final users.

Among the stakeholders usually the project manager is the key role and responsible for managing a project in all its aspects. The project management work involves the application of best practices, like methodologies, in order to meet project objectives.

The Project Management Institute (PMI) and the International Project Management Association (IPMA) are two important examples of organisations, in particular for what concerns standards, which are dedicated to the development of project management.
According to the Guide to the Project Management Body of Knowledge (PMBOK published by the PMI [PMI04]) project management activity is described as the application of knowledge, skills, techniques and tools in the activities of a project in order to achieve its objectives.

2.3. Information Technology projects

An IT project is a project with the objective to create a product or service, in which the utilisation of Information Technology is the decisive characteristic, the best example is the development of any software application.

The IT project is not only concerned with programming but there are others elements and processes associated, for example there usually are three processes that bring to a new system, similarly to other kind of projects, that are: [BH99, pp.2-3]

1. The feasibility study: investigation used for deciding whether a prospected project is worth or not. Information about general requirements are gathered. The probable developmental and operational costs, along with the value of the benefits of the new system are estimated.

2. Planning: if the feasibility study respond is positive, the planning of the project can take place. Depending on the size of the project the plan will be more or less detailed.

3. Project execution: the project is executed. As will be explained later in this stage, the project methodology has its strong influence.

But what mainly differentiates an IT project from other kind of projects is that the result is, since in most cases is software, is not tangible and often very abstract in its application. Another main difference is that the world of Information Technologies evolves at much higher rates than any other field, and through abstraction, this makes all the projects subject to probable changes.

As for all kind of projects also in the IT world there are different organisations working to define standards for project development. For example the Software Engineering Body of Knowledge (SWEBOK supported by IEEE Computer Society) [Soc04], that could be described as a software engineering version of the PMBOK treated before. The SWEBOK is an all-inclusive guide which represents the sum of knowledge within the profession of software engineering, not only software engineering but also other related
disciplines. It is composed by a series of Knowledge Areas that are explained for the related discipline:

- Software Requirements
- Software Design
- Software Construction
- Software Testing
- Software Maintenance
- Software Configuration management
- Software Engineering management
- Software Engineering Process
- Software Engineering Tools And Methods
- Software Processes and Product Quality

The SWEBOK has also been criticised in the past because it did not accurately reflect the community’s view of software engineering, and in addition it did not distinguish among possible roles within a software development effort. This does not express the most common practices in the field, where these distinctions are frequent. Different roles require different skills, which rely on different knowledge. But is at list is recognised the effort to try to define the entire software engineering profession. [DN00]

At the end it is possible to define a software engineering management as the application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software.

2.4. Success and failure

All projects need to be performed and delivered under certain constraints. There are three universal constraints that are followed for almost all projects: Time, Cost and Scope. These three constraints make up what is called The Project Management Triangle (Figure 2.1) and in the most of cases the respect of these constraints determines if a project will succeeds or fails. A further refinement of the constraints separates the
product "quality" or the "performance" from the scope, and turns quality into a fourth constraint.

Figure 2.1.: The Project Management Triangle

"The time constraint concerns to the amount of time available to complete a project. The cost constraint refers to the budgeted amount available for the project. The scope constraint regards what must be done to produce the project’s end result. These three constraints are often competing constraints: increased scope typically means increased time and increased cost, a tight time constraint might increase costs and reduce the scope, and a tight budget could mean increased time and reduced scope." [CC]

The discipline of Project Management pertain to provide the tools and techniques that enable the project team to organise their work to reach these constraints.

But why does a project fail? reasons may be different, and studies report different reasons [Cha03, p.26][Rub07], including:

- Cost and schedule are not reviewed during the project.
- Plans are not used correctly.
- Project managers are not trained with the necessary level of skill.
- The theory is viewed as a waste of time and not put in practice.
- The project scope changes.
- The incorrect methodology is used.
- Requirements have major changes.
2.4 Success and failure

- Lack of communications.
- Testing is poorly done.
- Lack of user involvement.

These are just some of the elements that can lead to failure a project, whereas a complete list would be difficult to achieve considering the uniqueness of each project.

The Chaos report of the Standish Group [Gro95] shows that among these factors of failure the one that seems to affect a project more than others is the user involvement. What are also reported as key elements for the success of a project the executive management support, a clear statement of requirements and an experienced project manager.

The Chaos report is the result of a research conducted in the United States involving various companies that deal with the development of IT solutions. One result of this research is the subdivision of projects, that the companies surveyed have undertaken since 1994, into three category succeeded, failed, and challenged. Challenged means that at least one of the three factors of success has not been came across, meaning that there has been a cost or time overruns or the result did not fully meet the user’s needs. [Gro95]

This statistic (Figure 2.2) would suggest that there is a process of improvement within the world of IT projects. The increasing number of projects that have been successful (from 16% in 1994 to 35% in 2006) and the simultaneous decrease of failed projects (from
2.4 Success and failure

![Graph showing cost of change over time for Traditional and XP methodologies.](image)

Figure 2.3.: Traditional and XP assumption about cost of change over time [Leff07]

31% in 1994 to 19% in 2006) may reflect an awareness by the project managers of the need to introduce some tools and methodologies that could have a role in improving the project management. But last data results show a remarkable decrease in project success rates, there has been a significant increase in the number of failures that represents the highest failure rate in over a decade, calling into question the past interpretations of this statistic.

As explained before the respect of basic constraints of cost, time and scope, is essential to the success of a project. The most significant element to define how much time, cost and effort will take to build a software, is the scope (or size) of what is to be built, and therefore one of the biggest factors to create an accurate estimation. Without a correct initial evaluation of the size of the project the risk of not meeting one or more constraint increases considerably.

Analysing another side of the triangle, for example the cost, it is possible to see that is strongly influenced by the other two sides. A factor of very great affection is given by the purpose that if it changes during the execution of the project he can have different effects depending on the methodology used. A change of requirements on a project that uses a classical methodology, for example the Waterfall, that will be introduced in the chapter 3, brings inevitably to a growth of the costs and also of the time necessary to end the project. As represented in the picture (Figure 2.3) these changes have a different weight depending on when are introduced. The change of requirements during the execution of a project and as the different methodologies react to them, it is one of the themes that will be treated in the next chapter.
In synthesis it is impossible to modify one of the sides of the triangle without having immediate consequences on the others. [CC]

The methodologies of work also try to give some general guidelines on to avoid, and in case happens, to manage all the problems that can be introduced during the execution of a project. Whether well organised and prepared or not, every project has to face sooner or later a problem which can be of any kind, from a human problem to a technical one. What changes from one project to another is the way they are addressed and solved. As will be explained later in the work is virtually impossible to plan every detail and anticipate every problem at the beginning of a project. [Cha03] One thing to beware of when it comes to projects, is the Murphy's law that is often quoted in the literature and in a pessimistic way states:

<Anything that can go wrong will go wrong."

But if on one side it is unlikely to foresee all the problems, it is possible to prevent and manage them, prefiguring all weak pixels (the "Anything that can go wrong" part) and try to consider all kind of solutions.

Each methodology has its own unique approach in solving problems.

A problem matrix example in Figure 2.4 shows that for any problem encountered during a project there is a cause and a related possible solution. In the picture for example there is the problem "over scheduled" that seems to be caused by the use of the waterfall approach and one possible solution proposed is to change the methodology choosing an iterative one, probably for the next project. [Cha03]

In the next chapter different methodologies will be presented in order to give a more
concrete perspective of diverse approaches used to conduct successfully a project, in order to reduce the risk of failure.
3. Introduction to methodologies

3.1. Introduction

This chapter will answer three research questions presented in the introduction which are closely interlinked:

- What is a methodology?
- What are the existing methodologies? How often are they used in IT projects?
- Which are the common points and differences between selected methodologies?

Software development could be defined as a chaotic activity, often based only on the coding work. In the past there was the habit of writing software without having a detailed underlying plan, and all the work was composed from many short term decisions. This can eventually work with a small project, but as the system to be built becomes bigger also the difficulty to add new features to the system grows.

This situation generates a movement dedicated to change this way to work introducing the notion of methodology. These methodologies impose a disciplined process upon software development with the aim of making it more predictable and more efficient. [Fox05]

In the world of IT project and in the literature related to this argument, there are different terms used for the concept of methodology. A methodology for IT project is also called simply method, or software development model, or also software engineering process.

But what is a methodology? The answer to this question can bring to different definitions, for example: [Cha03]

- A process that documents a series of steps and procedures to bring about the successful completion of a project.
- A defined process for accomplishing an end.
3.1 Introduction

- A series of steps through which the project progress.

- A collection of method, procedures, and standards which define a synthesis of engineering and management approaches designed to deliver a product, a service, or a solution.

- A integrated assembly of tasks, techniques, tools, roles and responsibilities, and milestones used for delivering the project.

In concrete a good definition, found in the literature, could be: "A methodology is a set of guidelines or principles that can be tailored and applied to a specific situation. In a project environment, these guidelines might be a list of things to do. A methodology could also be a specific approach, templates, forms, and even checklist used over the project life cycle" [Cha03].

What must be taken into consideration is the fact that a methodology is not a temporary solution or something that could be used for fixing part of a project which is not proceeding as intended. To apply a methodology takes time and above all it needs an adequate experience.

Methodologies found in the literature and in practice are different and can be defined with terms like classic, iterative, incremental or agile; these terms will be fully explained later in the work. Methodologies that can be called classics were first introduced and served as a development base for the others, for example there are:

- Waterfall [Chapter 3.2]

- V-model, considered as an extension of the waterfall.

- W-model

Almost all of the more recent methodologies are not as static as the classic ones, the development of projects is based on the iterative and the incremental phases. Among these methods there are some that has been called agile, in detail here there are some of them:

- Spiral model [Chapter 3.3]

- IBM Rational Unified Process (RUP) [Chapter 3.4]

- Agile methods [Chapter 3.6]
3.1 Introduction

- Dynamic System Development Method (DSDM)
- Adaptive Software Development
- Crystal Method
- Scrum [Chapter 3.6.1]
- eXtreme Programming (XP) [Chapter 3.6.2]
- Lean Software Development
- Feature-Driven Development

Another methodology that will be described is HERMES [Chapter 3.5], it is based on the classical V model, but its application in IT projects may become iterative and incremental, especially when applied with RUP as complement.

Nowadays one of the methodologies which attire the interest is certainly RUP, which has been used for more than a decade. In addition there is an increasingly attention to XP and Scrum which are part of the more actual methodologies called agile. There are no precise statistics regarding the use of different methodologies, but it is possible at least to classify the interest aroused by them from the articles, the case studies and general comments that can be found in the literature, or for example in the Internet. It must be taken into account that a large part of projects probably does not use any methodology in particular.

Almost all the recent methodologies have a number of key concepts that share in their definitions, such as: [JR05, p.176]

- **Phases**, series of steps needed to go though a project. Each methodology has different phases.

- **Milestones**, specific events that occur or items delivered during development.

- **Deliverables**, pieces of the final products that are delivered (code, documentation, demo, prototypes).

- **Schedule**, when the milestones must be accomplished.

- **Work assignments**, the members of the team are assigned specific tasks that are tracked during the development.

- **Communication** between all stakeholders.
Then how to choose between the different methodologies currently available? What are the factors that lead to choose a methodology rather than another? Some elements that can help in the decision are: [JR05, p.177]

- **Type of product**, depending on which kind of software will be developed a project could be bigger or smaller and the methodology used could be different.

- **Size of the team**, a small team can be successful with less formal methodology than a large one.

- **Type of people on the team**, some people work better with less formality, supervision, and planning than others.

- **Type of customer**, the customer may or not be available and actively participate during the project.

- **External constraints**, sometimes the use of a determined methodology is contractually imposed.

- **Past record**, if the past used methodology works maybe there is no need for change.

- **Simplicity**, large and complex methodologies might be more than a small team can handle.

In the next part of this introduction the concepts of standards and metamodel will be treated in order to give the necessary information for describing all methodologies.

### 3.1.1. Standards

As shown previously, there are large number of methodologies and ideas concerning the management of IT projects. Besides there are also many organisations which work on to establish standards of process or product for the software industry. The main purpose is to improve the quality of software products and their production processes.

The standards can be divided into two categories:

- **Standard de jure**: document "voted" by an assembly empowered to define standard. For example International Organization for Standardization (ISO) or Object Management Group (OMG).

- **Standard de facto**: standards that are "decided" by the market. For example PC hardware or Microsoft .doc format.
The main organizations involved in standards in the software world are:

- IEEE, mainly for methodologies
- OMG, for UML and CORBA
- W3C, for web technologies
- OASIS, for Business Process

An example of standardisation could be the Software Process Engineering Metamodel (SPEM) [Gro08] that is defined as a Profile (UML) by the Object Management Group. In other words it provides an extension mechanism for customising UML in the domain of software project methodologies.

SPEM is used to describe a software and the systems development processes and their components. The scope of SPEM is purposely limited to the minimal elements necessary to define any software and systems development process, without adding specific features for particular development domains or disciplines. The goal is to accommodate a large range of development methods and processes of different styles, cultural backgrounds, levels of formalism, lifecycle models, and communities. [Gro08]

OMG [Gro] is an international organisation that collects principal software sellers. His most famous standard is the Unified Modelling Language (UML) which is a visual notation used to describe softwares. A language modelling software like UML allows to specify, to visualise and to document a product and its development process. Models are documents that generally facilitate the communication between client and developers.

### 3.2. Waterfall

The first response to the software crisis, due to the lack of ability to develop correctly and efficiently the software for large application, was the discovery of the waterfall methodology in the late 1960s. It was thought that the main cause of the problem was that the requirements were not fully understood before pursuing to the other stage (design). Consequently the implementation was done with a poor quality design. [Joh]

Waterfall is considered the classical sequential methodology for software development. As shown in Figure 3.1, the waterfall is composed by a sequence of activities working from top to bottom. The arrows pointing downwards indicate the steps during the execution of the project. To correct eventual mistakes or lacks is possible to jump back in the previous step which has been introduced later. [Tec]
In the original waterfall model, the following phases were executed in order:

1. **Requirements specification**: detailed discussion of all users requirements of the system and a documentation of the proposed system is created.

2. **Design**: a two stage design that meets all requirements is created. One will represent what the system is to look like. The second reproduce the physical design, how the data and software procedure are structured.

3. **Construction (Implementation or coding)**: programming or modification of existing code.

4. **Integration**:

5. **Testing and debugging (Validation)**: testing that the proposed system meets the quality and requirements requested.

6. **Installation**: set up the entire system after the software development.

7. **Maintenance**: once the system has been installed there will be support for bug correction and eventually for extending and improving the system.

The procedure is purely sequential, for instance, when all requirements are completed the flow proceed to the design, afterwards it is implemented by programmers and continues to next stages.
One of the many critics about waterfall model is that it is quite unlikely to perfectly complete a lifecycle phase before moving on to the next. The Waterfall is a methodology that is considered inflexible, useful only if all needs are well defined at the beginning and do not change during the execution. Another problem encountered is that the testing phase is scheduled only at the end, and the product itself is available during the latest stages of the project. [DLP86]

But at least the waterfall model has served as a base-model for the future development of new methodologies.

3.3. Spiral

The Spiral model is a process originally developed by Barry Boehm [Boe88] to solve problems verified in the previous models (like waterfall). The Spiral model (Figure 3.2) integrates the concepts of project objectives, risk management, prototyping and project costs.

![The spiral model](image)

Figure 3.2.: The spiral model [Boe88]

The spiral model consists of four quadrants:
• Defining objectives, alternatives, and constraints

• Analysing and solving risks

• Developing and testing product

• Spiral Planning next iteration

Each cycle of the spiral model iterates through these four quadrants. The number of cycles is specific for each project, that is why the description of the activities in each quadrant are intended to be general so that they can be included in any cycle. [HG95]

With a good project overview, the objectives of a cycle can be decided. The following step in a cycle is to evaluate the alternatives and to identify and to resolve risks using tools like prototyping, simulation and benchmarking.

The next step in the cycle concern the development of all objectives and methods decided for it. In an earlier cycle the development might involve the evolution of concept of operation or software requirements, whereas in a later cycle it might mean development of software design, detailed design, or code. The development also involves a validation of the resulting output.

After it, the last part of the cycle is thinking about the work already done in the cycle and planing the next one. In earlier cycle, it might be define all requirements or the life cycle plan, whereas, it might be an update of earlier plans, development plan or integration and test plan.

Each cycle has an associated cost. The radial dimension of the spiral represent the project cost and the angular dimension represents the project progress for a cycle. [Boe88]

In conclusion, the spiral model allows to produce software until early stages during the project, and tries to ensure the quality through this prototypes releasing at the end of each cycle. From this point of view the spiral model was a big improvement over previous methodologies because it focuses on evolutionary incremental development, risk management, prototyping, and project overview and planning. This methodology remains very static and therefore less flexible, then it seems to prefer large projects with long iterations.
3.4. IBM Rational Unified Process

The Rational Unified Process (RUP) is a commercial process framework developed by Rational Software Corporation and now incorporated into IBM’s Rational Software Division. RUP development has been parallel with the Unified Modelling Language (UML) and incorporate parts of UML modelling constructs. Now RUP is also included into the IBM Rational Method Composer that will be described later in this work (Chapter 4.2).

RUP is based on a set of building blocks which describe what is to be produced, the necessary skills required and the step-by-step explanation of how goals are achieved. The main building blocks in RUP are the following:

- Roles (who) which defines a set of related skills, competences, and responsibilities.
- Work Products (what) that represents a result that has been produced by a task like documents and models.
- Tasks (how) that describe a unit of work assigned to a Role which afterwards provide some result. Task are categorised into different disciplines (Figure 3.3).

![Figure 3.3.: RUP overall architecture [Hau05]](image-url)
In the RUP process model (Figure 3.3) on the time axis there are four phases in the lifecycle, which have typically multiple iterations:

- **Inception**, the objective of this phase is to build the business case for the product and gain credits among the stakeholders. In this phase a business model, a vision document and a prototype could be produced.

- **Elaboration**, the project is planned, all features and the architecture are determined.

- **Construction**, all features are implemented onto the architecture using an iterative and incremental approach. Every iteration gives as result a usable piece of the code that can be tested and evaluated.

- **Transition**, the system is deployed providing all necessary migration support and finalising the user document.

Usually a project, depending on its size, can be divided in several lifecycle each one with a precise objective.

On the other axis of the model all the necessary software disciplines to achieve the result are represented. These include the six engineering disciplines: [Per03, p.40]

- **Business modelling**, the aim is to establish a better understanding and communication channel between business and software engineering. Business modelling explains how to describe a vision of the organisation in which the system will be deployed.

- **Requirements**, explain how transform stakeholder request into a set of requirements work products.

- **Analysis and design**, the goal is to show how the system will be realised. The design model serves as an abstraction of the code source.

- **Implementation**, describe how reuse existing components or implement new one with a defined responsibility, making the system easier to maintain, and increasing the possibilities to reuse.

- **Test** are carried out along four quality dimensions: reliability, functionality, application performance, and system performance. The process describes how to go through the test lifecycle.
3.4 IBM Rational Unified Process

- Deployment, successfully produce product releases and deliver the software to its end users.

There are also three supporting disciplines: [Per03, p.41]

- Configuration and change management deal with three specific configuration management:
  - Configuration that is responsible for the systematic structuring of the product.
  - Change request that keep track of the proposals for change.
  - Status an measurement attributes and store states for each change request.

- Project management focuses mainly on the aspects of an iterative development process like risk management, planning, and monitoring progress.

- Environment focuses on the activities necessary to configure the process for a project. It describes the activities required to develop the guidelines in support of a project.

The "dune" on the chart represents on different degrees how the various disciplines are exercised over time. For example in the construction phase all iterations will focus more on implementation, the test and the configuration than the other disciplines.

![Figure 3.4.: RUP’s iterative model](image)

RUP provides six essential best practices as online guidance for the most important activities in which a software development team is involved: [Lef07, p.54]
• *Develop iteratively* is the core of RUP (Figure 3.4).

• *Manage requirements* in a user-centric way. Implementation in RUP is driven by highest priority user.value use cases (one of the agile principle).

• *Use component architectures* like building blocks help a system achieve scalability.

• *Model visually* help the designers to reason about the system even when it became complicated.

• *Continuously monitor quality*, every iteration produce an increment of working code that must be tested. Continuous testing and test automation are integrated in RUP practices.

• *Verify change*, the only real constant in software development is change, RUP provides mechanisms to control and integrate change into software code base.

RUP tries to cover a very large set of practices, from business modelling to coding practices, to configuration management, and to the project management. This adds complexity and makes RUP less easy to be approached compared with other methodologies.

### 3.5. HERMES

Like for others countries the Swiss confederation aimed to have a standard methodology that could be used for every project inside its administration. The project management methodology HERMES has been used since 1975, with major revisions in 1986 and 1995, and is based principally upon the German V-Model.

Like other methodologies treated before, HERMES is a phase based on methodology (Figure 3.5) where some phases can be iterated. Activities and responsibilities are defined into each phase and also the necessary cross-tasks to the success of a project are grouped in sub-models.

The submodels describe the events, the activities and the roles according to a general approach to the project. For systems development practice HERMES provides in details the following sub-models: [HTT]

• Project Management

• Quality Assurance
3.5.1 HERMES and RUP

All sub-models do not depend on a defined type of project but they are used in all ICT projects. Typically a sub-model describes the various cross-cutting functions of a project, defining all roles and activities involved.

Even in HERMES, as in RUP, the projects are considered from three different perspectives (views), the first is the results produced, the roles on the project, and the procedures which define how the work is performed.

HERMES can be extensively tailored, using the PowerUser tool. HERMES is results-oriented and provides additional views on Roles and Procedures.

Since HERMES is considered a very general methodology that intend to deal with every aspect of a project, it becomes almost necessary to use a methodology for the part of the project which concerns the development of software. For this reason the use of RUP as a complement to HERMES will be explained in the next part.

3.5.1. HERMES and RUP

The relation between HERMES and RUP [3.4] has been treated in the manual [Swi03, pp.326-331] with the aim to provide all the information about the choice of the Rational Unified Process as complement model of HERMES.

The analysis is based on the comparison of the different metamodels such as methods (Figures 3.3 and 3.6) and a comparison of structural elements, such as phases, results, roles and work techniques.

The initialisation phase of HERMES does not exist in RUP because this process is only used for a project already defined. The finalisation phase of HERMES is included.
in RUP in the "close-out project" in the project management discipline. The remaining four phases are more or less identical, all activities, result and roles are essentially comparable.

The six HERMES sub-models correspond to the nine RUP disciplines. The system development, kept at a general level in Hermes, is more detailed in RUP and it includes business modelling, requirements, analysis and design, implementation and test. The quality assurance of HERMES in RUP is designed as an integral part of all disciplines. The marketing project has no correspondence in RUP disciplines because such activity is outside the engineering process.

The central elements, such as activity, outcome and role, are widely comparable in the two process models.

The conclusion of this analysis is that HERMES and RUP are ideal complements. The different concepts should be compared in a glossary. There are some overlaps in the activities and results which can be solved during the methodology tailoring and could differ from one project to another.

### 3.6. Agile methods

The term Agile Method has been coined in February 2001 by some leading developers and proponents of what were then known as "light" methodologies. [Koc05, p.3] The main agreement on the first meeting was the need for a method that was able to easily respond to the changements during the software projects. This ability end to adopt the term "Agile" for identifying those kind of methodologies. The following step was the
declaration of the "Agile Manifesto" composed by four statements that compose the core of all Agile methodologies: [AAV01]

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

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

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

These four statements are explained in more detail with the twelve Agile Principles in Appendix A.1.

All Agile methodologies have in common some aspects that touch topics like agility, change, planning, communication, and learning.

The term Agility is in contrast with some traditional software development methods which are considered more inflexible or static. Agile methodologies allow to react quickly to changes. These methods are designed not just to accept changes, but also to welcome them and capitalise on them. [Koc05, p.5]

On the other hand the traditional methods like waterfall treat the change as a risk, a possible cause of failure. Agile methods could be defined as adaptive compared to other classic methodologies which are predictive.

![Figure 3.7.: Plan-driven versus value-driven methods](Lef07, p.81)
Planning is a central topic in Agile methods like for any traditional methodology, the main difference is how deviations from the original plan are treated. The Agile method treats the deviation as a new information and creates a new plan based on the new information. The goal is to conform the plan with the actual situation (reality). The result is that all initial requirements estimated (Figure 3.7), the resources and schedule are fixed, this is the opposite of classic methodologies. As for changes, the traditional methods are less disposed to be adapted on builted plan.

Traditional method tend to assume that the communication between the various stakeholders is mainly produced by scheduled activities and all produced documents. While the Agile methods push less on the written documents in exchange of face-to-face communication, than on project team members and on the relation between the team and the customer.

The last point is that Agile methods treat a project as a learning experience. It must be taken into consideration the fact that every stakeholder in every project keeps on learning throughout the life of the project.

In the last 5 to 10 years, many new and different agile methodologies came into practice. They carried different names, tactics, activities, and acronyms, but they all aimed at the same problem: creating reliable software more quickly. These methods include: [Lef07]

- Dynamic System Development Method (DSDM)
- Adaptive Software Development
- Crystal Method
- Scrum
- eXtreme Programming (XP)
- Lean Software Development
- Feature-Driven Development
- OpenUP

In the next part of the work three Agile methodologies will be presented in more detail, Scrum, eXtreme Programming (XP), and an interesting new entry OpenUP, in order to give a more concrete picture of the field of software development methodologies.
3.6.1. Scrum

Scrum is an iterative and incremental methodology for software development used primarily with Agile practices. The name Scrum comes from the game of rugby and it refers to a strategy used to get a ball back into play [Koc05]. This is a metaphor for the development team that must work together in order to lead all stakeholders in the same direction.

It is based on three simple points: Sprint, Backlog and Scrum Meeting. It is very similar to Extreme Programming planning (Chapter 3.6.2), Scrum divides the project into fast work blocks (Sprint) at the end of which it deliver a version of the product to the customer, it shows how to define the details of work to do in the future (backlog) for a broad definition, and it organises daily meetings of the development team (Scrum Meeting) to verify what has been done and what will be done.

![Figure 3.8.: The Scrum process model [KS02]](image)

The main roles in Scrum are: [Lef07, p.42]

- *Scrum Master* that maintains the processes and works similar to a project manager, is responsible for the success of Scrum. The primary responsibilities are to:
  - Ensure that all Scrum directives are followed.
  - Identify who will take the role of Product Owner.
- Facilitate all Scrum practices
- Act like a communication interface between management, customers and the team

- **Product Owner** that represent the stakeholders. Give to the Scrum team a business perspective of the work and writes down User Stories to place in the Product Backlog with a established priority.

- **Team** that includes the developers and have the responsibility to deliver the product. Usually is a small team of five to nine people.

The term Scrum Team comprehend all three roles that are also called "Pig" roles based on a joke. [Sch04]

A pig and a chicken are walking down a road. The chicken looks at the pig and says, "Hey, why don’t we open a restaurant?" The pig looks back at the chicken and says, "Good idea, what do you want to call it?" The chicken thinks about it and says, "Why don’t we call it 'Ham and Eggs'?" "I don’t think so," says the pig, "I’d be committed but you’d only be involved."

In the others side there are "Chicken" roles that are not committed to build software regularly and frequently, whereas they must be involved in order to provide useful feedback for reviews and to plan each sprint. In particular there are:

- **Users** for which the software is written
- **Stakeholders (Customers, Vendors)** that will enable the project. They are directly involved only during sprint reviews.
- **Managers** that set up the environment for the product development organisation.

During a Sprint, a period between 15 and 30 days, the team create an increment of potentially usable software. All features implemented in each sprint are described in the product backlog, which contains with a priority order all the work to be done. The backlog items are treated in a specific sprint is determined during the sprint planning meeting. During this meeting the Product Owner claims which parts of the product backlog it wants to be completed. Based on those information the team decides how much of the work committed can be executed during the next sprint. Once the sprint starts, the sprint backlog and all requirements are frozen until the next one.
"Software development is a complex process that requires lots of communications. The Daily Scrum meeting is an event where the team comes to communicate" [KS02]. During the Daily Scrum, each team member answers at three main questions: [Sch04]

- What have you done since the last Scrum?
- What will you do between now and the next Scrum?
- What got in your way of doing work?

This is a short meeting of about 15 minutes which takes place every day in the same place.

"Customers, users, management, the Product Owner, and the Scrum Team determine the next SPrint goal and functionality at the Sprint Planning meeting. The team then devises the individual tasks that must be performed to build the product increment" [KS02].

The outputs of the Sprint planning meeting are the Sprint objective to be achieved during the Sprint and the Sprint Backlog.

At the end of a Sprint cycle, a Sprint retrospective (of about four hours) take place with the purpose of improving the process. The two main questions discussed are:

- What went well during the sprint?
- What could be improved in the next sprint?

This is done in order to provide all information about the progress situation and it is the base for the next Sprint Planning meeting.

There are mainly three documents that are produced using the Scrum methodology, in detail there are: [Lef07, p.45]

- Product backlog is an high level document for the entire project, it resume all the work that has to be done from major features to a wish-list items. The Product Owner is the main responsible of prioritising all backlog items.

- Sprint backlog is a more detailed document that contain all information about the requirement that will be implemented during the next sprint. Usually all tasks contained in the Sprint backlog are not greater than 16 hours work.

- Burn down chart (Figure 3.9) display the number of tasks remaining for the current Sprint. Is a visual and publicly displayed representation of the Sprint backlog.
One of Scrum's advantages is that it is quite easy to learn and it requires only a little effort to start using it.

3.6.2. eXtreme Programming

Extreme Programming (XP) is one of the most popular Agile methods. XP has been described by its creator, Kent Beck, as "a lightweight methodology for small to medium-size teams developing software in the face of vague or rapidly changing requirements". [Bec00]

XP similarly to other methodology focuses on three main roles according to them some key rights and responsibilities, like "pig" roles in Scrum for example, in detail there are:

- **Customer** that chose what will deliver business value, what must be done first and which tests the software needs to pass.
- **Manager** bring customer and developer together and help the collaboration.
- **Programmer** analyse, design, test, program, integrate and estimate the time necessary to deliver stories.

The detailed description of the bill of rights for all roles could be found in Appendix A.2, these rights give a good base definition for all roles in XP projects.

As shown in Figure 3.10 the XP lifecycle is composed by iterations of two to three weeks each leading to short releases every two or three month, up to the major release every six to twelve months.
XP is based on the application of twelve guidelines, these practices are its defining features:

1. **User stories** (Figure 3.11) are a short description of the behaviour of the system from the point of view of the customer. They are used to estimate the necessary time for release planning and they are planned at each release and short releases. They must be also chosen and ordered at each iteration and could be modified during the lifecycle. Each story has at least one or more acceptance test defined. Once the story is created the developer assigns an estimated risk and a cost value.

2. **Small releases** and iterations are part of the highly iterative development process of XP. A short release cycle is usually up to 3 months and is composed by several cycle that are up to 3 weeks. A user story should be completed in few days.

3. **Simple design** ensures that there is less to communicate, to test and to refactor. The programmers must use the simplest possible design which will allow the current story to be implemented.

4. **Tests** play an important role in XP, they should be written before the code is developed. There are two kind of test:
• Acceptance tests (functional tests), customers provides test cases for their stories.
• Unit tests, developers write tests for their classes.

5. *Refactoring* is the process of improving code while preserving its function. Refactoring is used to ensure that the design remains simple and does not grow into complexity. All tests are used to guarantee that refactoring does not break anything that worked.

6. *Pair programming* is the most visible practice of XP. Two programmers sit together in front of a workstation, one enters code and the other reviews it. It is considered as the second most important practice after tests. Pairs form and change dynamically throughout the project according to the needs of each story.

7. *Metaphor* defines the key objects and interactions for the solution, it also designate a common set of names and interfaces between classes and systems. This is a very high-level vision of the project. The stories with the project’s metaphor form the requirements from which the system is planned and developed.

8. *Collective code ownership*, the code does not belong to a single programmer but to the team. A programmer can change the code at any time in order to test it and to make it simpler and better.

9. *Continuous integration*, the entire system is built frequently. When a story is completed and the code is integrated the entire suite of test is executed. Only if all tests pass, the story becomes part of the project baseline.

10. *Forty-hour week*, programmers should not work more than one week of overtime, if this is not the case then something is wrong with the schedule.

11. *On-site customer*, customer is considered part of the team and should be always available to resolve ambiguities, set priorities, and provide test cases.

12. *Coding standards*, let the team setup coding standards before coding and agree on choosing common name choosing scheme and common code formatting. In a team with a so closely collaboration is important to maintain standards to assure the order.
In the traditional software methodology the lifecycle begins with listen to the customer for gather all requirements, then design all objects that are added to the system. The next step is to code all objects and at the end all tests are executed. XP make a substantial difference switching the development of test cases from the end and putting it just after the listen phase.

The philosophy behind XP is well resumed by the following citation: "Software development is too hard to spend time on things that don’t matter. So, what really matters? Listening, Testing, Coding, and Designing." [Bec00]

3.6.3. OpenUP

In the past years RUP has become one of the most widespread methodology in software organisations. However RUP has also a complexity and a learning curve which can bring some difficulties to implement it in all organisations.

In 2006 IBM donates some of RUP property to the Eclipse Foundation. The part gifted was the basic Unified Process, a reduced version of RUP.

The most agile and lightweight form of OpenUP (OpenUP Basic) targets smaller and co-located teams, which are interested in agile and iterative development. Small projects constitute teams of 3 to 6 people and involve 3 to 6 months of development effort.

OpenUP is a lighter weight process with only six roles and 18 tasks. As RUP, it remains iterative, incremental, use case-driven, architecture-centric, and still follows the lifecycle phases of RUP, besides it incorporate some principles and practices from others agile methodologies like: [Lef07, p.70]

- Scrum, product backlog and iteration planning, assessments, and daily status meet-
3.6.3 OpenUP

<table>
<thead>
<tr>
<th>OpenUP Key Principle</th>
<th>Agile Manifesto</th>
</tr>
</thead>
<tbody>
<tr>
<td>Collaborate to align interest and share understanding</td>
<td>Individuals and interactions over process and tools</td>
</tr>
<tr>
<td>Evolve to continuously obtain feedback and improve</td>
<td>Responding to change over following a plan</td>
</tr>
<tr>
<td>Balance competing priorities to maximise stakeholder value</td>
<td>Customer collaboration over contract negotiation</td>
</tr>
<tr>
<td>Focus on articulating the architecture</td>
<td>Working software over comprehensive documentation</td>
</tr>
</tbody>
</table>

Table 3.1.: Similarities between OpenUP and Agile Manifesto

- Agile modelling, no big upfront design
- XP, test first, refactoring and continuous integration
- DSDM, stakeholder collaboration

As shown in table 3.1 OpenUP, it is based on four core principles that have a correspondence with the Agile Manifesto.

OpenUP could be seen as an hybrid of RUP and a number of agile methods, both very popular at the moment.

OpenUP projects are executed over a series of iterations (Figure 3.12), typically two up to six weeks in duration, comparable with the Scrum practice. Iterations are started
with a short planning activity where the highest-prioritised items, from the work items list and risk list, are allocated to the iteration plan. This is followed by a short activity where the team gets involved in the detailed planning and in the estimation of each work item. Thereafter the iteration work commences and each feature is analysed, designed, implemented, tested and integrated in its own micro-increment. As the iteration unfolds, task status is reported back to the iteration plan for overall project status. [Gus08]

The project lifecycle (Figure 3.13) identifies four phases which are the same that could be found in RUP: Inception, Elaboration, Construction, and Transition. This lifecycle differs from agile methods focusing on the inception and on elaboration phases. It comprehends the scope of the project and its solution, that covers approximately 30-40% of the project schedule with a 20-25% of the budget spent.

The OpenUP process features practices suitable for many projects and also provides the basis for adding extensions using the EPF Composer tool, allowing the tailoring of the most appropriate process for each project.
4. Tailoring with methodologies

4.1. Introduction

This chapter will answer the question presented in the introduction: How to do "tailoring" of methodologies?.

Very rarely software development projects adopt a methodology in a rigid way. Methodologies are normally adapted to meet specific contextual characteristics. This adaptation is known as Tailoring.

Tailoring is an activity that can be done in different ways, normally by taking a methodology and choosing only the parts that can be really useful to the project, but it also possible to take parts of different methodologies.

So after the explanation of methodologies, which can be very theoretical and abstract, it comes to implementing them.

This implementation can be done in a manual way, doing some best practices with the creation of diagrams, tables, graphs, and all artifacts that are necessary to guide the project development; or by using some frameworks, or software, that allow IT teams to create a very detailed guideline of the project which will subsequently be used to direct and oversee the work.

There are a lot of software products with the aim to better manage an IT project in all its aspects, including scheduling, cost control and budget management, resource allocation, communication, quality management and documentation.

But these programs individually, from the least to the most complex, serve as support, but do not help to implement a methodology of work within a project.

Concerning the practical application of a precise methodology [Foub] three solutions based on the Eclipse platform will be presented here:

- IBM Rational Method Composer with the RUP methodology
- Eclipse Process Framework with OpenUP and other agile methods
- HERMES Power User
4.1.1 Unified Method Architecture

These solutions contain all the elements and best practices of methodologies, and can be used as a starting point for companies and development teams for trying to implement a working method, and to create a discussion on how to tackle a project. They also provide all the tools needed to modify, adapt, and create from scratch a methodology applicable to a given context.

4.1.1. Unified Method Architecture

The Unified Method Architecture (UMA) metamodel has been developed as a unification of different method and process engineering languages such as the Software Process Engineering Metamodel (SPEM) extension to the UML for software process engineering. It provides concepts and capabilities from all of these source models unifying them in a consistent way, but still allowing to express each of these source methods with their specific characteristics. [Fouc]

![Figure 4.1.: UMA concept of method content and process separation](Hau05)

The UMA metamodel provides a language for describing method content and processes. UMA is an architecture to conceive, specify, and store method and process metadata. UMA separates Method Content definitions from their application in delivery processes by defining the reusable core Method Content, in the form of general content descriptions such as Roles, Task, Work Products and Guidance and the project-specific applications in the form of process descriptions. Basic elements of UMA are shown in Figure 4.1.
The UMA metamodel create the possibility of a strong customisation and of facilitate the tailoring work.

The UMA metamodel has been used as base of the Rational Method Composer and the Eclipse Process Framework that will be presented in the next part.

4.2. Rational Method Composer

The IBM Rational Method Composer is a commercial product, defined as process management tool platform for authoring, tailoring and deploying development processes for IT projects. It is a platform, built on top of Eclipse [Fou01], for process engineers, projects leads, and project and program managers which are responsible for maintaining and implementing processes for development organisations or individual projects. [Hau05]

The mainly requirements for the development of a similar platform is the need of having a flexible development processes in order to support tailoring and to make all processes evolve across projects. This platform is composed by a set of tools that allows the creation and configuration of method content and processes.

There are two goals that RMC want to achieve. The first is to provide a knowledge base for developers. RMC is a content manager that gives a general management structure for all contents that could be next published in HTML and be deployed into the Web. The second goal is to provide the necessary support for process engineers and project managers in selecting, tailoring and assembly processes to become concrete development project through catalogues of adaptable processes for typical situations. It also provides some capability patterns (process building blocks) representing development practises for specific disciplines, technologies or development styles. [Hau05]

Rational Method Composer, like the Eclipse Process Framework, has the aim to provide solutions to general problems encountered when acquiring and managing methods and processes. For example:

- Is important to have a centralized access to the information, processes need to be deployed and accessed at the workplace.

- Difficulty to integrate different proprietary format for development processes.

- Teams lack of base knowledge on methodology and best practises.
4.2 Rational Method Composer

- **Teams support for right-sizing their processes** that need to be tailored continuously through the project lifecycle.

- **Ensure compliance to standardised practices.**

- **Effective execution of processes in project**, bridge the gap between process engineering and process enactment.

The most important outset for RMC is the separation of application processes from the core method content as described into the RUP process model (Figure 3.3). The method content describes what will be created, the skills required, and how developments goals are achieved. Processes, on the other side, describe the development lifecycle. This separation has been already presented in the RUP methodology from its creation.

Many methodologies are described with a step-by-step description for a particular way of achieving a specific development goal under general circumstances. RMC takes content and structure from the UMA schema, all method contents are represented in RMC as a construct of roles defining development skills and responsibilities for work products. These work products are created by tasks that are performed by the roles and have work product as inputs and outputs. [Kro05]

In contrast with methods, development process defines sequences of how is being performed by roles and how work products are being produced and evolved over time. RMC support processes based on different development approach with different lifecycle models like waterfall, incremental or iterative. It is also possible to define processes which use a minimal set of method content or no methodology content at all.

As shown in Figure 4.1, methodology content is defined using work product, roles, tasks and partially guidance. On the other side there are activities that are related to each other in order to define a work flow. The delivery processes that represents a complete template for executing a specific type of project and the capability patterns that are processes that express process knowledge for a discipline or a best practice.

RMC could be described as a container of method and processes the necessity is to select, configure, and tailor the existing content in order to satisfy the needs. All documents are not necessary for a single project, this makes the tailoring process a very important step for organising the framework.

Some of the typical usage scenarios of working with RMC but also with EPF are:

- **Selecting and configuring existing method content and processes**, this is the simplest
way to use those platforms by selecting process and method content that fits with a specific need by browsing the method library.

- Tailor an existing process, in addition of configuring a process is possible to actively modify processes to make them more adapt with specific needs.
- Create a new process, an alternative to tailoring is to create a complete new process from scratch or reusing parts of existing processes.
- Develop method content and create or extend processes, develop a personal method content and use that with existing process or created one.

There is an open source version of the IBM RMC that is part of the Eclipse Process Framework project. There are some differences between EPF and RMC especially for what concerns integration with other Rational tools. EPF comes with OpenUP, an agile process for small teams that apply RUP principles and practices. EPF will be treated in the next part of the work.
4.3. Eclipse Process Framework

The Eclipse Process Framework (EPF) is an open source project, built on the Eclipse platform, where it is possible to collaborate around software practices. It can also produce and deliver some examples of practices or processes that can be used as a starting point for different organizations and project teams. It also has some underlying infrastructures and some tools helping to create and to customise content. This infrastructure is extensible, so that it is possible to build different extension tools on the top of it (Figure 4.3).

The Process Framework Project has two goals:

- To provide an extensible framework and exemplary tools for software process engineering - method and process authoring, library management, configuring and publishing a process.

- To provide exemplary and extensible process content for a range of software development and management processes supporting iterative, agile, and incremental development, and applicable to a broad set of development platforms and applications.

By using EPF Composer it is possible to create a personalised methodology for software development by structuring it in one specific way following the UMA schema previously presented.

Per Kroll in an interview [Inf07] describes two cases of the use of EPF. In the first one, a project team may decide that it needs to get some ideas on how to follow various iterative approaches, so it gets some guidance and can take EPF, download it, and look for what is available there. It is possible that the teams could find interesting parts, this will provide several ideas on how to improve them, and thus it can be modified and adapted to the situation. Another usage model would be when an organization says that it would like to have some standard approaches for software development and build up a common language for communicating around software development environment.

The main output of EPF Composer is a web site, or a sequence of web pages, which provide guidance to software developers and track the work.

The idea with EPF is to build libraries electronically, indicating best practices for software development. Organisations can easily tailor it to their needs by mixing and matching processes quickly for all projects.
An example can be found in Appendix B where Figure B.2 shows the application EPF Composer, which in the left pane represents all the available content of the methodology chosen (in this example it is scrum) and in the right box the work product "sprint_down_chart" is opened and can be modified as required. In this case it has been simply taken the graph presented in the chapter 3.6.1, and replaced the default one. The picture below (Figure B.1) shows the result of the changes on the web pages generated by EPF Composer that can now be easily shared between the work team or the organisation.

In conclusion EPF is currently a tool that allows to pick out practices and publish an HTML website that will be used like a reference library for a project or an organisation.

4.4. HERMES PowerUser

HERMES PowerUser is an Eclipse Process Framework based on the implementation, it supports the execution of all HERMES projects. This tool, as the previous cases, contains all the tools and guides that allow to put in practice the methodology.

It supports both project versions System development [Swi03] and System adaptation
4.5 Considerations

[Swi05] and guides the user from initialisation to conclude all ICT projects. HERMES PowerUser give the possibility of: [fIT]

• Browse the documentation of both types of projects.
• Manage the results as Word documents.
• Export the work steps to Microsoft Project.
• Adapt HERMES to specific projects (tailoring).
• Define customised process and generate document models.

The program, as in the case of EPF, is downloadable for free directly from the website [fIT].

4.5. Considerations

The use of these frameworks can be very useful in several ways. It will leave the set of all the components of one or more methodologies within a single software. These elements can be simply used as a starting point in the discussion and decision process that anticipate the beginning of a project.

In a more advanced way it is possible to choose parts or subsets of these methodologies that can be applied to an environment or a specific project making any necessary changes to create a customised working method, including considerate that some practices are very hard to express because they emerge only in best practices.

Finally it is possible to describe these frameworks as big boxes tightly organized and extensible, from which all the materials necessary to guide the development of software projects are available.
5. Case studies

5.1. Introduction

This chapter will answer the question presented in the introduction: Which are the positive and negative experiences of IT projects that use methodologies?. Two case studies will be presented that are related to the practical development of IT projects. These case studies will give a more defined vision of the use of methodologies in real projects. This part of the work can be defined as the practical one, opposing to the previous chapters which were more theoretical and related on the existent literature.

The first case study that is going to be introduced in this chapter, is the one of SiSO SA company. This case study has been created using the material provided by the company and by the interview made to Mrs. Borislava Dumont, IT developer and project manager, Mr. Marko Marjanovic and Mr. Luca Genasci, two IT developers of the team.

The second case study that is going to be presented is the "all e-dec" project, that is a project of the Swiss administration, which is divided in 4 subprojects. This case has been realised thanks to the material and the information provided by Dr. Stefan Hüsemann, supervisor of this work, whom holds the role of project management and requirements engineering inside one of the subprojects; the e-dec export (IDEE) project.

The typology of the two case studies that will be discussed is very different. The first case is an overview of the work methodology within a small company that deals with the development and the sale of its own platform on the market. Around the platform several minor projects are developed in order to improve and adapt to them the internal needs and demands of customers.

The second case study treat a single project of medium-large size and complexity conducted by the Swiss Federal Customs Administration in order to replace old administration solutions with a new one.

An outline of the interviews is available in Appendix C.
5.2. Case study: SiSO SA

The interviews and the material collected serve to resume and to have a global vision of the concept of the work methodology for IT projects.

From the beginning, the firm is analysed in its whole, its purpose and its products. Subsequently it is considered a specific product in which the interviewed has had the opportunity to direct as project manager; in this case the whole platform proposed by the firm.

Departing from these information the organisation of the company is described in general. Afterwards it is shown the way the team devote itself to the project, in particular how it is composed and how organise the work during the execution of the project.

In the principal part of the case study the use of methodology is analysed, from the phases which compose it to the most serious problems found during the projects.

Subsequently the decisional process of the projects are crossed. The management of the work inside the team, like the relationships with the clients and the requests are treated.

Finally the matter of the use of software and specific work documents for achievement of the projects and also all the positive and negative considerations of the use of the actual methodology of work are treated.

Company description

SiSO SA is a company founded in 2004 by Lombardi Engineering Ltd., SD Ingénierie Holding SA and Amberg Engineering AG. The design, development and application of prime IT solutions and operations in support of construction projects is the main purpose of the company. [Sa]

The company mission could be resumed in six points:

- Provide software applications of considerable help throughout the life of a construction project: from the planning, during the site, up to the operation phase.
- Coordinate all the parties involved in large civil engineering projects.
- Provide a common platform for the management of information at every level.
- Support the day-by-day construction management and supervision, including real-time control of costs, schedule and quality.
- Promote the concept one data input for many users on site.
## 5.2 Case study: SiSO SA

### Module Description

<table>
<thead>
<tr>
<th>Module</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SiSO-a</td>
<td>Base module.</td>
</tr>
<tr>
<td>SiSO-b</td>
<td>Module for the supervision of construction works, provided by the Site Diary.</td>
</tr>
<tr>
<td>SiSO-c</td>
<td>JAVA-based graphical user interface for data acquisition, consultation and plotting.</td>
</tr>
<tr>
<td>SiSO-d</td>
<td>Geological module.</td>
</tr>
<tr>
<td>SiSO-e</td>
<td>Monitoring module.</td>
</tr>
<tr>
<td>SiSO-f</td>
<td>Controlling module (construction schedules and costs).</td>
</tr>
<tr>
<td>SiSO-g</td>
<td>Module for special contractor’s data.</td>
</tr>
<tr>
<td>SiSO-h</td>
<td>Module for the damages to works.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table 5.1: SiSO modules</th>
</tr>
</thead>
</table>

- Encourage to store the "Story of the Construction", for present and future purposes.

### SiSO Platform

SiSO is a commercial integrated management tool for construction site supervision, designed to meet the specifically requirements of all the parties involved in large civil engineering projects. SiSO has been particularly suited for linear projects, such as underground works or bridges (Figure 5.1).

This software is primarily used in engineers offices and within the management of large construction sites.

This platform has a modular structure that allows to be specifically used in different project for different purpose (Table 5.1), for example:

- Project management.
- Construction documentation archiving.
- Day-by-day works validation by contractor and site supervision.
- Management of construction schedules and outstanding activities.
- Standardised sharing of up-to-date and selected information.
- Continuous support to construction supervision.
- Periodical update of costs estimates and schedules.
5.2 Case study: SiSO SA

- Online access to construction progress and results.
- Advanced plotting and report generation features.

SiSO could be defined as an Oracle database Web application. It is developed in PL/SQL, the Oracle's extended query and update language, and now is progressively switching to PHP. The software is composed of:

- database in Oracle
- procedures, functions and packages in PL/SQL and PHP
- files (css, javascript, images, ...) on the web server

To give some concrete examples here are four practical projects, amongst various, in which the software in question was used:

- Gotthard-Base-Tunnel, 5 Lots, 153 km, investment of 8000 million CHF
  - 110 platform users, Modules SiSO-b, -c, -d, -f, -g
- Lötschberg-Base-Tunnel, 3 Lots, 88 km, investment of 1700 million CHF
  - 50 users, Modules SiSO-b, -c, -d, -f, -g
- Refurbishment of the interstate A12 (P3), 13 km, design and construction, investment 100 million CHF.
  - 30 users, Modules SISO-b, -f, -g
- Métro Lausanne, 6 km, construction, investment 80 Million CHF
  - 10 users, Modules SISO-b, -f, -g

Organisation and team management

The company currently has eight employees whose four are in the team which deals with the development of the SiSO platform. There are also external collaborators who are contacted from other companies depending on specific needs.

SiSO SA currently has two offices one in Fribourg, where almost the entire development team is located, and the other in Minusio which are also the customer and project responsible and a member of the team.
5.2 Case study: SiSO SA

(a) The project record (diary) (SISO-b)

(b) Comparison of the actual performance with the performance according to the contract (tunnel)

(c) The graphic interface allows the integration and one time storage of the data. (SISO-c)

Figure 5.1.: SiSO platform - Planning (a), Construction (b) and Operation (c) phases
In group there is a responsible for server and database management and the other members have more or less the same knowledge to continue the work if someone is on vacation for example. Since it is a web platform, the most part of the work is exclusively web engineering; to be more precise: various procedures pl/sql for the Oracle database, some php, and java applets.

**Projects description**

First we must pin down the term project within the company. SiSO platform can be considered as a major project which has a development over time due to many smaller projects which are divided into two main categories (Figure 5.2).

On one hand there are projects inside the company, in addition the head of research and of development is usually responsible for constantly monitoring and proposing innovations which can be applied to the software. Afterwards the most interesting developments are chosen by the heads of the company.

On the other hand, there are external projects. For what concerns the customers sales, the person in charge for the coordination is generally responsible to help the project leader of customers to install (in most cases the hosting is done by SiSO company) and configure the software.

![SiSO projects organisation](image)

**Figure 5.2.: SiSO projects organisation**

The project itself is when a customer with the purchase of the software has a specific request for one or more applications in the platform. In this case a specification plan is
created and therefore the development of the project will be started.

Among the various internal projects the company has also tried to subcontract some of those to companies in Bulgaria. After some projects carried out, it was decided not to pursue outsourcing because it has been found more and more difficulties for what regards the respect of time and budget constraints.

**Methodology used**

In the past the company has tried to use some programs and methodologies but considering the tiny size of the development team, it was decided that the application of a specific methodology was not necessary.

For a tactical decision the SiSO software itself was chosen to manage all projects, in particular for the subdivision of the work, for establishing deadlines and for creating comments on the various tasks (Figure 5.3).

In practice, the process that follows an external project, starts from a professional engineer, which shall discuss all specific needs with the customer.

Thereafter, the engineer is responsible to create and submit a business concept, which summarises how the application must function with respect to the needs of the customer.

Subsequently, the business project is analysed by the development team, taking into account all the possible IT solutions to the problem.

Once the work is established and divided, the development phase begins. Depending on the size of the project, the development can be divided into different steps that lead to the evaluation tests which are usually performed by the one who created the business concept. Based on the tests results, the project is adapted to the changes and new possible requirements.

To summarise, the development of the projects it uses a methodology that can certainly be described as iterative and have several phases of development and testing. This is due to the fact that often the development team must retrace its steps in the case that customers realised that they had forgotten something, or wish to change the initial requirements.

One consideration based on the experience done in the past years, that has been made during the interview, is that it is virtually impossible to be able to have all the data and the precise request at the beginning of a project because they often tend to change during development.

For what concerns the management of budget and time, during the planning phase, some estimation of necessary work hours to complete the project are made. Because
the work is almost entirely based on programming, the hours of work also represents the budget, or the estimated cost, of the project. Then depending on the project, and especially on the needs of customers which can change during execution; these estimations are met or modified during the development.

From the experience gained from several projects, a large number of them do not get to reach the initial estimations of time and budget because of the many changes in the requirements during execution. This however, seems that does not create any major problems, since the changes of specifications (reflected upon the budget and time) are proposed and discussed by the customers themselves.

So far the company has had no external projects that have failed, at least there were only slight delays which are not quantified as serious if they go from one to a maximum of three weeks.

On the other hand, for what regards the internal projects, there is more freedom in managing the work that must be done. These projects are not a priority, they must not be used immediately on the field but they are made to keep the software updated with the latest technologies, to increase the value and the usability of the platform for future internal and external projects.

Because of the tiny dimension of the team and of the realised projects, the produced documentation and the software used to complete all projects is limited. Firstly the code is commented and then thanks to the utility phpDocumentor, the help documentations are created for internal use. Also UML is used for the graphical representations of parts of the platform, as was recently done with the finances module of SiSO. For the portrayal of the database an utility called toad, which provides a graphical vision, is used to facilitate the development of new solutions.

There are other useful tools which are used during the projects, Subversion (svn) manages different versions of the platform and it is also planned to introduce an application which will allow to do all automatic update, it will send automatically patches to the hosting servers.

Methodology considerations

The first point made which concerns the working methodology used to manage project, treats the problem of relocation of a member of the team. This relocation brings slight disadvantage in terms of communication within the team. Give that, as was said before, the development team is small and relies heavily on teamwork also daily meetings, the fact of not having a team member may bring some problems in managing the daily
5.2 Case study: SiSO SA

(a) archive and quality control

(b) diary (journal) and tasks

Figure 5.3.: SiSO test development platform
tasks of work. This disadvantage is reduced with the use of Skype in order to maintain a stable contact and have some meetings on a regular basis. The obvious advantage of having the whole team in the same place is to allow a better solution of problems through teamwork, which in this case becomes an individual work.

Another consideration made is that since the SiSO platform has used for civil engineering projects, there are some disadvantages and limitations during the IT projects. An example proposed is that when the tests are realised there are some difficulty to find problems and bugs depending on how the code is structured. In other words, the platform does not provide a support for the debugging work. SiSO platform is not conceived for IT projects and is not prepared to follow specifically and to find possible bugs in the code.

Another aspect may be that the lack of a program of works, it is a graphical display that helps to trace the state of the IT project.

These are two possible areas of software development for company internal projects in order to give an IT project orientation at the SiSO platform.

As final consideration of the interview, the major problem of project management has resulted in the choice between reducing the work into many small units and then check them all the time or give more freedom to developers. The problem is that if the work is reduced in too small units, it is like directly doing the work. In the other side giving too much freedom is hazardous, since the project might take a wrong way from the one intended. The organisation of work is the most difficult task of projects.

5.3. Case study: e-dec

E-dec is an Information System created for the Swiss Federal Customs Administration (FCA). Customs clearance of goods (cargo processing) is one of the core responsibilities of the FCA, in fact every good that enters or exits in Switzerland must be declared by companies.

Because of the political, legal, technological, and economic developments of the past few years, the Federal Customs Administration has now a very broad palette of products at its disposal, all of which have the same objective: the customs clearance of goods. These products include diverse form-based and IT-based solutions for the import, transit, and export of goods.

This multiplicity of cargo processing products increasingly results in service offerings which are perceived, both by clients and by the Federal Administration, as too broad
and unclear. Moreover, the cargo processing products offered are often isolated solutions that are not sufficiently harmonised. This leads to efficiency deficits and higher costs for all parties involved.

All e-dec project aims to remedy these problems. With e-dec, the Federal Customs Administration is developing a central cargo processing product that will gradually replace existing products, such as the electronic customs clearance system M90 and the form-based transit procedure.

![e-dec architecture](image)

**Figure 5.4.: e-dec architecture**

E-dec has a modular structure. In this way, the FCA aims to offer clients the possibility of performing electronic clearance of import, transit, and export goods using largely uniform guidelines. In this way, more uniform and streamlined processes are created for both sides, with the result of lower costs. [Adm] [EldA08]

As shown in the Figure 5.4 the architecture of e-dec is service-oriented (SOA), and offers various services to the outside but it is itself composed of a series of services. There are several layers of communication and integration (Enterprise Service Bus) from which the services are made available to the outside.

The institutions that are involved with the project "all e-dec" are different, from the
FCA instructing the execution of the entire project to the Federal Office of Information Technology to Systems and Telecommunication (FOITT) that realises all IT parts. Also all the various departments that receive data from the software are involved, for example, the Federal Office of Public Health (FOPH) which receives data on imports and export of medicines. Other end users, along with Swiss federal departments, are the customs officers and thousands of enterprises engaged in import-export outside the borders of Switzerland.

**Organisation and team management**

The "all e-dec" is divided into 4 sub-projects (Figure 5.5), e-dec Transit, e-dec Import, e-dec Infrastructure, and e-dec Export (or IDEE). The latter will be discussed in more detail in this case study, because the interviewee is its project manager.

The project can be defined as medium-large, it has a budget between 1 to 5 Million Swiss Francs, it is run over three years, and it involves 20 employees for what concerns the development of the IT part.

The project e-dec export, for its part, has an average of 12 employees. Within the team there are some specialists, in particular system architects are involved to ensure that all sub-projects are consistent and compatible. Others key roles are the test manager, the requirements engineer in addition to the role of developer.

![Figure 5.5.: all e-dec projects organisation](fddA08)
The IT development team for the project e-dec export is located entirely in one floor of the same building. This provides a major advantage at the level of communication and coordination. The ability to have direct contacts, not only for what concerns the project, it gives more benefits. In the event that a problem related to the projects occurs, the responsible can be reached immediately for establishing a possible solution. Although some prefer to work directly at home, the contact and direct confrontation between all employees creates a better working environment for the project.

During the execution of the project from time to time some sub-teams (with a sub-team leader) are put on place for the development of some specific use cases, a set of features, which are linked together. These sub-teams follow these cases from specification to implementation.

Methodology used

For what concerns the methodologies used, Hermes is the official one for all projects within the Swiss public administration. As was previously explained (Chapter 3.5) this methodology is very general and it deals with every aspects of a project into its smallest details such as, for example, how to sign a contract, or the initialisation phase of a project, deal with issues and practises that are carried out before even thinking about the project itself.

Hermes is therefore used throughout the e-dec projects, in the production and the customs side, and for communicating between all stakeholders. But Hermes is not specific enough to follow the development of software. For this reason the part of IT development of the e-dec export project uses RUP which allows to have a better suited methodology for the development of IT projects.

The two methodologies are compatible, as explained above [Chapter 3.5.1], and the different phases are linked together.

In the first phase of RUP (Inception), which corresponds to the Preliminary Analysis phase of HERMES, the project is defined. All Business Use Cases are elaborated and then grouped in a business model. Every Business Use Case is analysed in details and must be accepted by a group of specialists, in this case, the customs officers.

Each use case contains a particular activity (function) to be examined. All use cases, which form a package, can be assembled in a field of functions and activities.

In the second phase (Elaboration), which corresponds to the Design phase, all the use cases and actors in the system to be built are defined. A system use case describes the interaction of users with the system. All system components and all the relationships
between the various components are described. The architecture helps to have a vision of the system to be built, showing the design of the technical solution. In RUP the architecture is documented using UML diagrams according to different view and several documents or artifacts are created and used during the project (Figure 5.6).

In the third phase (Construction), which corresponds to the Implementation phase, firstly the design of the system is formed, then a prototype is created and afterwards the entire system is realised. Finally the introduction of the system is prepared and all necessary protective measures are put in place. By following all the various iterations related to the RUP methodology, the system in all its phases is continually tested and implemented step by step.

Contrary to what is called in Hermes, for the e-dec project prototypes are produced already during the design phase. Every month a release of the product is published and is available for testing by Customs officers.

In the fourth and final phase (Transition), which corresponds to the last two stages of HERMES Deployment and Finalisation, when, as in the case of e-dec, the new system replaces an old one and it is possible to distinguish two basic ways to proceed:

1. Big Bang: the existing system is completely replaced at once with the new system.
2. Step by Step: the existing system is gradually replaced by the new system.

Because of the importance and the complexity of the e-dec system, it was introduced
using the procedure Step by Step.

Initially the e-dec export project as described in general terms and afterwards the realisation of time which compose the various lifecycle is estimated.

The project e-dec export is divided into 3 RUP lifecycle of the duration of 1 year each. Every lifecycle has a main precise objective, the first is to have a pilot, the second is to get a product that is already functioning, and the third is to have the finished product.

As the project is executed, the phases that make up a lifecycle (inception, elaboration, construction, and transition) are described in more detail. From an overview of the entire project, as the project progresses, it switch in to a much more detailed one.

Different iterations are planned for each phase and it is estimated the time needed to perform each task, and each month every member of the development team will receive a list of tasks to perform. These tasks are assigned with 3 different levels of priority. Level A makes the execution of the task required, while B suggested, and C optional.

During the execution of an iteration each team member manages its own tasks, for this reason the effort is firstly estimated in order to see if the work plan is realistic or not.

If any problem results with a task of level A, the project manager is immediately informed in order to better organise the work, perhaps adding the necessary workforce to be able to reach all the objectives.

At the beginning of each week there is a meeting where each team member states which tasks assigned to him wants to complete during the week. At the end of the month (end of iteration) another meeting is done, where the work done is summarised, it is determined whether the project follows the planning, if all the tasks has been completed, and thus, whether the iteration has achieved its goals in order to define the next iteration.

Finally it takes place a discussion on how to improve or change something in the way the work is to be done and on which the methodology adopted. These improvements are then taken in account during the execution of the following iterations.

For what concerns the documentation the RUP web pages are used with a work of tailoring to execute the project.

Methodology considerations

The use of the RUP methodology concerning the development of the IT project and the subproject e-dec export has yielded good results. The use of Hermes for the IT part is not possible as it exclusively treats all aspects surrounding the IT part of the project.
In any case, the communication between all stakeholders is made by using Hermes while for the internal processes in which IT developing is involved, it is used RUP.

There are no particular limitations for the utilisation of the RUP methodology for project management, RUP is applied optimally, but the only aspect which is emphasised is the difficulty to transfer the methodology outside the team, for example by influencing others to implement it. For this reason RUP methodology remains internal to the IT development team of the e-dec export project.

For testing the e-dec export software, three companies have given their willingness to follow it since its first release. There have been problems in finding companies interested in the experimental stage, the fact is that almost all companies wanted the final software, and cannot understand the utility to be a tester and to be involved right from the beginning of a project. The initial versions and prototypes are not particularly well-accepted, as many companies expect that everything works perfectly regardless of the version of the software they receive.

The risks associated with the project were well managed and the project is being completed according to schedule. Before the project begins, a risk list was created to be able to anticipate all possible problems and related solutions, that could arise during the execution.

A problem, that occurred after the end of the first lifecycle, was the increase in the
number of members of the IT team and the change of six of them which has led to a sort of internal training in order to make operational as soon all elements of the team as possible.

Another problem was that despite the companies had the task of adapting their software to communicate properly with e-dec, they found themselves unprepared. The problem arose from the fact that most companies do not conceive the development of IT solutions in an iteratively way. In fact, they waited for the software e-dec reaching its final stage before starting with the development of their software. This is in contrast with the iterative approach that provides as fast as possible some intermediary releases, so that companies can develop it in parallel.

There were no particular changes in the constraints during the execution of the project, which gave a certain stability. As for the timing of implementation, the only problem was the lack of preparation of the companies at the introduction of the software, nevertheless they have obtained a postponement of one year in order to better prepare the transition.

5.4. Case studies Analysis and conclusions

The case studies here introduced try to represent two different methodologies included in the world of IT projects, among the numerous ones that exist. Therefore it is not possible to analyse and to conclude if adopted methodologies are effective or not, but the single cases can show problems and peculiarity in comparison to the methodology procedures represented in the literature.

The SiSO case study can be interpreted as the situation in which a precise methodology is not used to conduct the projects, since the size of the projects and the IT team are limited. But this is not a total lack of methodology, but rather the use of a tested routine of work that introduces different elements of various methodologies picked in the existing literature.

For what concerns the management of the tasks inside the team, a brief meeting is done every day before beginning the work, during which the point of the situation and the work to perform during the day is established. This practice is very similar to that described in the methodology Scrum as Daily Scrum Meeting where in 15 minutes every member of the team responds to three questions. The negative aspects in this routine is that it approaches a discordant element if a member of the team is not present on the spot. Therefore he cannot use the reunions as method of contact, but has to work entirely with daily telephone contacts and through messages in the development platform.
that is also used for dividing all tasks through the team.

![Figure 5.8.: Representation of the SiSO SA Projects lifecycle](image)

Another interesting element is the use of representations (UML artifacts) to describe parts of projects, as describe inside RUP methodology for the analysis and design disciplines, even if it is not a consolidated practice for all projects.

For what concerns the contact with the clients, who also are in the greatest part of the cases the final users of the product, there is a certain distance with the development team. In the agile methodologies is underlined the fact that the final users must be deeply involved with the development of the project. In this case study is the "Customers and Projects" responsible, and he acts as connection between the clients and the IT team and he covers what can be defined as project initialisation phase. Also he deals with all evaluations test during the execution of the project, analysing with the client if the software satisfies all the specifications agreed, or if further developments and changes of requirements are necessary. Once that all new specifications are defined the "Customers and Projects" responsible entrusts the IT team to analyse and to develop them.

Considered that the initial specifications inevitably change during the execution of the project, the development lifecycle (Figure 5.8) maintains an iterative process involving several phases, as represented in all modern methodologies. As it has previously been described, to the phase of initialisation follows then three phases, which can perform different iterations, that is analysis of the specifications, implementation and functional tests (or debugging), and evaluations tests. Crossing more times these three phases the
final version of the product has reached and then deployed.

The second case study that was presented initially focuses on the methodology adopted for the entire project all e-dec, in this case HERMES. Afterwards were described all details and issues related to one of the four sub-projects of all e-dec: e-dec export. Ultimately, it was introduced in detail the methodology RUP that has been used for the development of the IT part of the sub-project.

Compared to the first case study on IT development teams and the size of the project does not allow the use of only work routines to lead the project. For this reason RUP and HERMES are fully implemented, even for what concerns the part of IT development. Therefore RUP remains to a degree of flexibility that allows to modify the methodology according to the need.

Other interesting aspects of the methodology adopted are the meetings which are conducted weekly to define the objectives that each team member wants to achieve. The meeting at the end of each iteration allows to summarise what has been done, what can be improved, and after that schedule the next iteration.

All stakeholders are fully involved in the project even if some problems, as described above, occurred with the companies that are also part of end users of the software.

In the Conclusions the RUP methodology has produced excellent results, thanks to its iterative processes and to a certain degree of flexibility which has allowed to well manage the e-dec export. Using a different methodology, such as an agile one, it is difficult to implement because of the size of the development team. In any case, some typical practices of agile methodologies are used depending on needs.

To conclude this general analysis about the two case studies presented, it can be said that these are two opposite situations. The first case describes the methodology of work of an entire company, SiSO SA, in which the main project is the platform with all his modules, which is also the only product offered along with hosting services. All IT development efforts are therefore around the various modules that make up this platform, through the development of small to medium projects, conducted by a small size team.

The other case study, e-dec, presents a single medium-large project within a public administration. The development team and the IT project budget is considerably larger than in the first case study. Hence also the methodology, although partly imposed by the administration, is more precise and applied in a systematic way.

The methodologies adopted in the two case studies, here represented, allowed to reach the goals set by individual projects.
A project is a unique activity that is undertaken in a given period, has a purpose and constraints that define it. Every project has risk in it that can lead to the failure of the project. IT projects, like any other project, are subjected to these risks. The discipline of Project Management has the goal of minimising the possibility of failure, providing knowledge, and practices to the execution of projects. Survey data conducted within the world of IT projects lead to think that over the past 15 years there has been an improvement in the techniques that can lead to a project success, even through the margin of improvement is still great.

Between knowledge and practices developed over the years some are defined as methodologies. A methodology is nothing else than a series of best practices, processes, and utilities used to improve the management of a project. Certain practices can be very easy and intuitive, such as writing a "to do list" of the work to be done, and others are more complex and require experience to be put into practice.

Of all the methodologies presented the current state-of-art prefers those incremental and iterative. This iterativity allows, in the field of software development, to have a greater flexibility and freedom of action in order to work out possible problems and manage risks during the execution of a project.

What almost all existing methodologies have in common is the part of the content (work product, role, task), and differ mainly in what concerns the processes, and how those contents are organised.

It is very interesting to follow the current evolution of these methodologies, in particular those called Agile, which allow the application of a work philosophy that focus on communication between all stakeholders and an extreme flexibility in case of changes in the requirements of the project. Alongside these Agile methodologies there are also others, such as RUP, that have an established best practices and experience over many years.

Chapter 4 has allowed to introduce some software that can be used to insert, and implement methodologies within projects and companies. Those platforms have the
advantage of containing entirely a methodology, from all components to best practices. This allows to select, modify, or create from scratch, parts of methodologies to adapt to the needs of the team, to the project, or to the entire organisation.

The two case studies presented have provided a vision of the application of a methodology within real projects. In the first case (SiSO SA) it is used no official methodology, but rather apply a series of well-established practices created with the experience with past projects.

But this does not create a static environment, there are always new ways to improve the working methodology to be tested, like the introduction of new kind of documentation, or to improve the web platform used to manage projects. The software development maintains its iterative characteristic that remains the most important.

The management of communication is the most delicate part of this case study because the company is divided into two sites distant from the other, and this leads to difficulties in the involvement of all stakeholders within the different projects.

The second case study (e-dec) represents a situation that is almost the opposite, or at least very different from the first. This is a single medium-large project, conducted by a governmental entity, it is subjected to the use of a precise methodology for the whole project (HERMES), but which allows the integration of the RUP methodology as a complement in the IT development. The case presented provides a picture of how a methodology is applied and what this means in the reality of daily work.

The main limitation of this work is that it is not possible to assess the effectiveness of the use of a methodology than another. But is likely to evaluate the overall context of the different methodologies found in the literature and also from the best practices. With case studies is possible to bring practical examples of how these methodologies are used within projects.

There are several interesting topics that can be explored, including for example the analysis of the effectiveness of a methodology rather than another using the analysis of case studies, but in a more significant amount, or by following the path of changes in the working methodology in a company.

Another interesting point would be to choose and to apply one of the methods described to a project, and then follow the evolution, and drawing a personal experience.

For what concerns the platforms for the tailoring of methodologies, in particular the Eclipse Process Framework, it would be interesting to analyse more thoroughly all the possible uses, and the future development inside implementation of all methodologies related.
This work has allowed me to improve, and deepen my knowledge about the management of IT projects. In particular, which are the methodologies currently available, and what tools exist to accomplish them. The experience of creating the case studies gave me the opportunity to know how projects are managed in practice, and in different situations. These cases also permitted to join the most abstract and theoretical part of the work to a more practical and concrete.
References


Referenced Web Resources


Referenced Web Resources


71
A. Appendix: Introduction to methodologies

In this appendix are presented the manifesto of agile methodologies and the Bill of Rights of eXtreme Programming. These two statements are intended to establish guidelines for the development and the application of these agile methodologies.

A.1. Agile Manifesto principles

Principles behind the Agile Manifesto [AAV01]

We follow these principles:

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

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

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

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

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

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

Working software is the primary measure of progress.

Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
Continuous attention to technical excellence and good design enhances agility.

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

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

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

A.2. eXtreme Programming Bill of Rights

"Extreme Programming tries to provide certain benefits to the managers, customers, and developers involved in a project. We express these as rights because they are very important to the success of the project, and to the team members." [RJH00]

Manager and Customer Rights

• You have the right to an overall plan, to know what can be accomplished, when, and at what cost.

• You have the right to get the most possible value out of every programming week.

• You have the right to see progress in a running system, proven to work by passing repeatable tests that you specify.

• You have the right to change your mind, to substitute functionality, and to change priorities without paying exorbitant costs.

• You have the right to be informed of schedule changes, in time to choose how to reduce scope to restore the original date. You can cancel at any time and be left with a useful working system reflecting investment to date.

Programmer Rights

• You have the right to know what is needed, with clear declarations of priority.

• You have the right to produce quality work at all times.
• You have the right to ask for and receive help from peers, superiors, and customers.

• You have the right to make and update your own estimates.

• You have the right to accept your responsibilities instead of having them assigned to you.
B. Appendix: EPF

This appendix presents two screenshots of one possible operation using the Eclipse Process Framework. In this example the methodology Scrum is imported into EPF, and the work product "Burndown chart" is customised (Figure B.2). Following the methodology is published and the corresponding web pages are created (Figure B.1).

![EPF web pages - Artifact: Sprint Burndown Chart](image-url)
Figure B.2.: EPF Composer authoring tool
This is a general summary of the questions asked during interviews for the preparation of case studies. These questions have the goal of touching all the topics regarding the methodology used for project development. The interview questions for SISO SA case study are based on the methodology used within the company and not for a specific project, as is the case for the second case study.

**Project/Company in general**

- What is the mission of the project/company?
- What are the services offered by the product/company?
- Can you describe the size of the project/company?
  - What is the number of staff/employees?
  - What is roughly the project budget?
  - What is the time for the development of the project?
- The project is currently running?/Do you have projects running at the moment?
  - Is respecting all the constraints imposed?
- Who are the end users of the product?
- What are the different kinds of projects conducted?

**Stakeholders**

- What is your role within the project/company?
- How is composed the staff in the project/company?
  - How many members make up the IT development team?
  - What are the specialists needed for the implementation of the project?
C Appendix: Interview schema

- What are the institutions/partners involved, directly or indirectly with the project?
- You do some kind of outsourcing for what concerns part of the project?
- Can you describe the location of the IT team/staff?

Methodology

- What methodology is applied to the development of the project?
  - You apply a single methodology for all projects?
- Can you give me a general description of the work methodology?
- What about work within the development team?
  - Meetings are conducted?
  - How is work divided?
- How are the end users/customers involved in the project?
- How the constraints control is maintained?
- What documents and work utils are used in the management of work?

Considerations

- Which are for you the advantages and disadvantages of the current methodology?
- There is the possibility of using HERMES for the part of the IT project? Do you want introduce a methodology in the future?
- Which are for you the advantages and disadvantages of the documents and work utils in use?
- There were problems during the execution of the project?
  - The methodology was changed during the execution of the project?
  - There have been changes in the constraints?
  - Have you had projects that have failed?
- Do you have other considerations regarding the methodology used for the development of IT projects?
Je, soussigné(e), déclare sur mon honneur, que j’ai personnellement préparé le travail qui précède et que celui-ci est conforme aux normes de l’honnêteté scientifique.

J’ai pris connaissance de la décision du Conseil de Faculté du 09.11.2004 l’autorisant à me retirer le titre conféré sur la base du présent travail dans le cas où ma déclaration ne correspondrait pas à la vérité.

………………………………, le …………………………………20……

………………………………

(signature)