ISCTE IUL
Instituto Universitário de Lisboa

Department of Information Science and Technology

Applicability of Agile Methodologies in Global Software Development Projects - A Scrum Case Study -

João Pedro Diogo Sampaio

A Dissertation presented in partial fulfillment of the Requirements for the Degree of Master in Management of Information Systems

Supervisors:
Professor Doutor Mário Romão, Professor Auxiliar, ISCTE
Professor José Cruz Filipe, Professor Auxiliar Convidado, ISCTE

Co-Supervisor:
João Barreiro, Agile Coach

September, 2011

This page intentionally left blank

Index

Table Index

Figure Index

Abbreviations

APO	Area Product Owner	PBL	Product Backlog
CASE	Computer-Aided Software Engineering	PBR	Product Backlog Refinement
CoP	Community of Practice	PO	Product Owner
DoD	Definition of Done	PoC	Proof of Concept
E2E	End-to-End	ROI	Return on Investment
F2F	Face-to-Face	SME	Subject Matter Expert
GSD	Global Software Development	SP	Story Point
GSE	Global Software Engineering	SoS	Scrum-of-Scrums
GUI	Graphical User Interface	US	User Story
LoC	Lines of Code	VoIP	Voice over Internet Protocol
PAPO	Proxy Area Product Owner	XP	eXtreme Programming

This page intentionally left blank

Abstract

Agile methodologies are based on co-located, self-organized teams, close and constant communication. On the other hand, Global Software Development raises a significant number of challenges related with communication, social-cultural differences and geographical dispersion. Therefore, applying Agile methodologies in a Global Software Development is not a straight forward practice and raises several challenges that need to be efficiently addressed in order to reach success.

In this dissertation is presented a study about how Agile methodologies can be successfully applied in Global Software Development.

A literature review is first presented in order to understand the state of the art about the theme of applying Agile methodologies in Global Software Development. Understanding which studies are currently available in the scientific community and their findings about either challenges faced and strategies applied in the studied projects.

A case study about a real project, where Scum is applied in Global Software Development, is also presented. Understanding how the Scrum practices are applied in the project, identifying the challenges faced and the strategies to efficiently address those challenges are the main study goals.

Keywords: *Global Software Development, Agile development, Scrum, Project Management*

This page intentionally left blank

Resumo

As metodologias Ágeis de desenvolvimento de software são baseadas em equipas co-localizadas e auto-organizadas onde a comunicação é constante e próxima. Por outro lado, o desenvolvimento geograficamente distribuído de software levanta um número significativo de desafios relacionados com a comunicação, com as diferenças sócio-culturais e a dispersão geográfica. Assim, aplicar metodologias Ágeis de desenvolvimento de software num ambiente geograficamente distribuído não é uma prática simples e levanta uma série de desafios que necessitam de ser eficientemente contornados para se garantir o sucesso do projeto.

Nesta dissertação apresentada-se um estudo sobre como metodologias Ágeis de desenvolvimento de software podem ser eficientemente aplicadas num cenário de desenvolvimento geograficamente distribuído.

Assim, inicialmente apresentada-se uma revisão da literatura no sentido de se identificar qual o estado da arte sobre o tema da aplicabilidade de metodologias Ágeis em desenvolvimento geograficamente distribuído. São identificados quais os estudos que se encontram atualmente disponíveis na comunidade científica sobre o tema e quais os seus resultados relativos aos desafios identificados e às estratégias aplicadas nesses projetos estudados.

Posteriormente apresentado-se um estudo de caso relativo a um projeto real, onde a metodologia *Scrum* é aplicada num cenário de desenvolvimento geograficamente distribuído. Os principais objetivos são os de perceber como as práticas *Scrum* estão a ser aplicadas no projeto, assim como o de identificar quais os desafios verificados e as estratégias aplicadas para eficazmente contornar esses desafios.

Palavras Chave: *Desenvolvimento de software geograficamente distribuído, Desenvolvimento Ágil de software, Scrum, Gestão de Projetos*

This page intentionally left blank

1 - Introduction

With the economy globalization and an extreme competitive business environment, companies are driven to constantly improve effectiveness and efficiency as well as customer satisfaction. To face these challenges, software companies started applying Global Software Development practices, distributing its development activities over different global locations using off-shore, outsourcing, subcontracting and partnership strategies. Also, to address the constant evolving customer requirements and quick time-to-market needs, companies are swapping traditional waterfall software development with more recent Agile methodologies.

Global Software Development raises a significant number of challenges related with communication, social-cultural differences and geographical dispersion. Agile methodologies are based on co-located, self-organized teams, close and constant communication. Therefore, applying Agile methodologies in Global Software Development projects is not a straight forward practice and raises huge challenges that need to be efficiently addressed in order to be successful.

Due to a non-disclosure agreement, required by the performing company, all information about the company and product used for the case study in this dissertation was kept as from a generic source.

1.1 - Motivation

The dissertation theme was selected mainly based on two motivations:

1. Its current relevance in the scientific and corporate domains. The interest in understanding what is the most recent knowledge within the scientific community about how to efficiently apply Agile methodologies in global distributed software development.
2. To provide a contribute to the company, where the study was performed, with an analysis on a specific software development project distributed over different sites, reaching conclusions that can be used in future to improve projects' efficiency. Furthermore, the same results can contribute to the scientific community to increase the available empirical knowledge on the theme and contribute to future knowledge consolidation.

1.2 - Problem

As introduced before, the applicability of Agile methodologies in Global Software Development is not straight forward and far from being a solved theme. The geo-distribution raises several challenges that have impact in the efficiency of the development process.

Therefore, the problem proposed to be addressed in this dissertation was the identification of the critical success factors to efficiently apply Agile methods in Global Software Development. An answer to this proposed problem was more specifically addressed in the two research questions next described in section 1.4 -.

1.3 - Dissertation Objectives

The main dissertation's objective was to understand what are the challenges raised when applying Agile methodologies in geographical distributed software development as well as what are the best practices and strategies to efficiently overcome those challenges.

In order to achieve this objective, a study about the current state of the art in this theme was performed in order to understand how far is this subject studied and consolidated. Then, a case study about a real geographical distributed software development project was performed in order to acquire additional empirical knowledge that could be compared and appended to the already existing in the community.

The expected result of this dissertation was to achieve a set of consolidated common challenges faced when applying Agile methodologies in geographical distributed software development as well as a set of best practices to efficiently overcome them.

1.4 - Research Questions

The broad questions addressed in this research were:

- *What are the challenges faced when applying Agile methods in a Global Software Project?*
- *What are the strategies to address the impact of those challenges in the software development's project efficiency?*

1.5 - Research Methodology

In this section, the research methodology used in this dissertation will be presented and justified.

The study proposed in this dissertation was about (1) figuring out how can Agile methodologies be successfully applied in a distributed environment and (2) trying to answer to research questions in the form of “what” questions. The main focus was to acquire empirical knowledge by observing the evolution of a real and recent event in its real-life context, without interfering on it. The literature review was performed aiming to get benefit from previous studies to guide the data collection and analysis.

The problem addressed in this dissertation was complex and diversified. As described in the literature review chapter, there are several questions that the literature classifies as not answered or not scientifically proven, thus recognizing that the available knowledge about the application of Agile methodologies in a distributed environment is scarce.

Although research questions were explicitly in “what” form, the dissertation also tried to answer to questions in “why” form. It was not only tried to identify phenomena, but also to provide an explanation for them. This was tried to achieve via the interviewees’ experience as well as by providing an interpretation of the observed and measured phenomena during the project.

According to Yin (2003) a case study is an empirical method used to investigate a contemporary phenomenon in its own real context without requiring any behavioural control of the events. Furthermore, the same author refers that case studies use multiple sources of evidence and may be based on previously developed propositions to guide the researcher during the data collection and analysis processes. Regarding the form of research questions, Yin (2003) refers that “what” form of questions can fit in any research strategy. Runeson et al. (2009), commenting on the collected data types, refer that a case study tends to be mostly based on qualitative data, but that whenever a combination of qualitative and quantitative data is possible, a better understanding of the phenomenon to be studied can be acquired.

The exploratory case study, with the utilization of the “what” and “why” form of questions, was therefore selected as the appropriate research method to deal with a problem where the lack of comprehensive knowledge is extensive.

It was possible to use multiple sources of data. Although qualitative data was the main data type collected, quantitative empirical data were also retrieved namely in which concerns the development performance metrics.

1.5.1 - Unit of Analysis

The subject of the case study was a large software product development project, globally distributed over multiple sites where Scrum methodology was applied. The product in question was divided in three components, each of them being developed by a different development unit. The unit of analysis in this study was one of these product's components, named in this dissertation "Component 3". The Component 3 development unit had its development distributed over four sites: two sites in Portugal (Site1; Site2), one in Poland and one in India. The main development site was Portugal Site 1. The other two product components (Component 2 and Component 1) were being developed, one in India and the other in China.

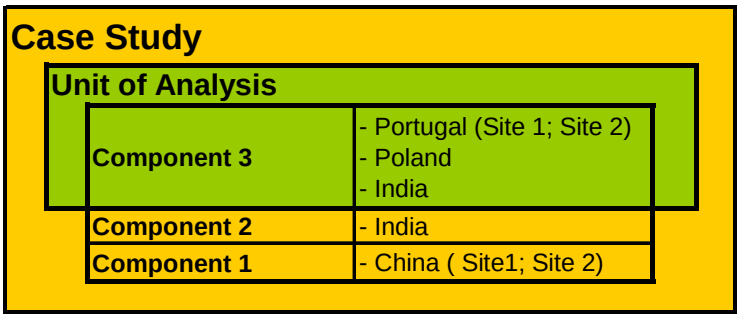


Figure 1 - Case Study and Unit of Analysis

1.5.2 - Data Collection and Analysis Techniques

The following data sources and analysis techniques were used in the case study:

- Observation and document analysis:
  - to get information about how project and teams were structured;
  - to understand how Scrum practices were being applied;
- Development performance metrics analysis:
  - to measure development performance evolution along the project;
- Semi-Structured Interviews:
  - to seek information from project's participants about their perception regarding the challenges faced, when applying Scrum in a distributed environment.

1.5.3 - Research Process

In order to reach dissertation objectives' and to find answers to the research questions the process presented in Figure 2 was followed and will next be described:

1. **Literature Review**
   - The first step was a literature review in order to frame the theme and present the most recent investigation results in the area;
   - Literature Review focus:
     - Evolution path from traditional waterfall models to Agile methodologies;
     - Overview of Scrum method, as it is the Agile methodology used in the project in study;
     - Evolution from co-located to distributed software development, what where the drivers and benefits that have lead software companies to start distributing its development projects over different locations in the world;
     - Collecting information about the challenges faced in Global Software Development and the strategies to face those challenges;
     - Scientific papers review about the applicability of Agile methods in Global Software Development.

2. **Case Study**

   2.1. Case Study Design and Planning
   - In this step the case study design and planning were performed.

   2.2. Data Collection
   - This step started with an iterative observation period of the Scrum practices applied in the project;
   - Semi-Structured interviews were designed and planned.
   - Semi-Structured interviews realization.

2.3. Information Analysis

- All data collected during the data collection step were organized, selected and analysed.

3. Summary and Conclusion

- Dissertation's conclusions and answers to the proposed research questions.

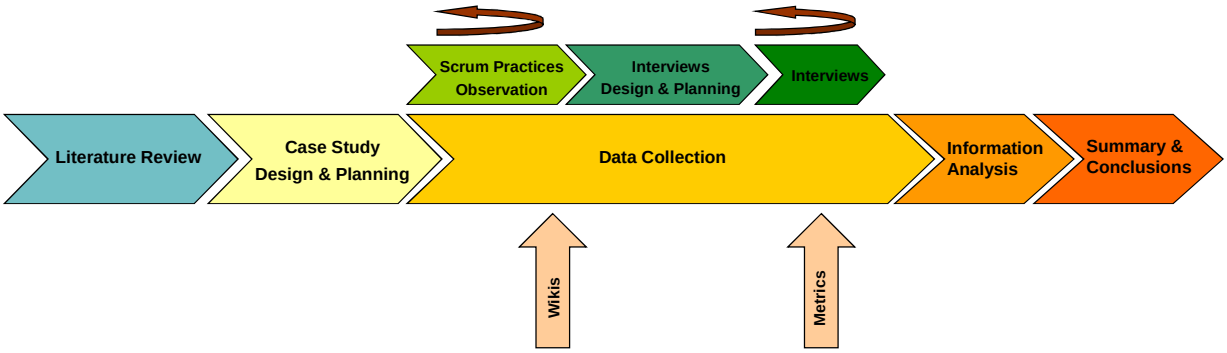


Figure 2 - Case Study Research Process

2 - Literature Review

This chapter will present the literature review findings and results about the state of the art about the theme in study in this dissertation.

The review will follow the evolution path from the traditional waterfall models to the recent Agile methodologies, as well as the evolution from co-located to global distributed software development. Special focus will be held on the Scrum method description, as this was the Agile methodology used in the project studied in this dissertation. Regarding the applicability of Agile methods in Global Software Development, the results from a review about the current available scientific papers will be presented. There will be found, a description of the type of studied projects as well as a consolidation of the most current knowledge about the challenges faced in applying Agile methods in Global Software Development and strategies applied in projects to successfully address the challenges. This chapter ends with a synthesis and conclusion about the current state of the art in the theme in study.

2.1 - Traditional Software Development

Traditional methods of developing software follow a sequential life cycle of phases. These methods are known as Waterfall models and typically are divided into five phases: Requirements, Design, Implementation, Testing and Maintenance.

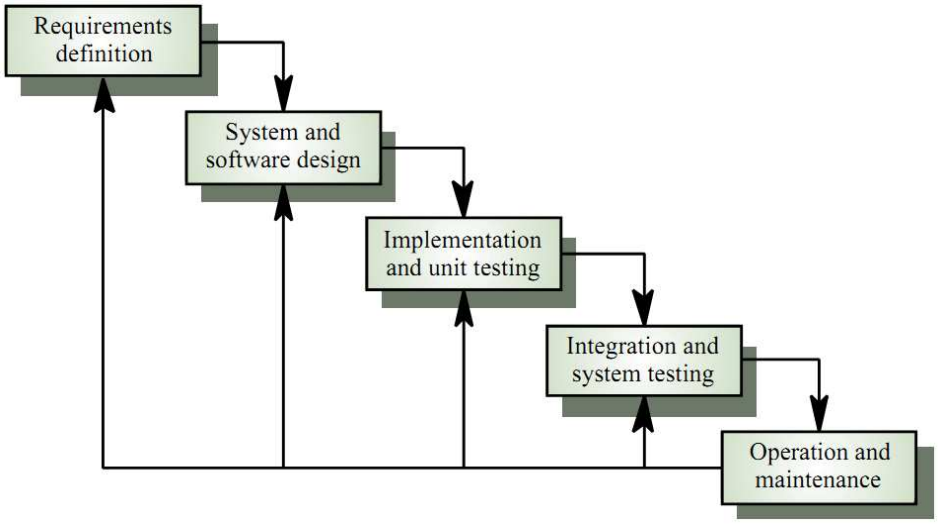


Figure 3 - Waterfall Model (Sommerville, 2007)

The production process starts with the Requirements phase where the attempt is to define and document in detail all or most of the product's requirements before programming it. The work throughout the different production phases is planned in detail, typically using Gantt charts. All the teams involved read the requirements specification and come to an estimate of how much will cost and will last their individual tasks until the end of the process. The detailed plan and the estimations are reviewed and approved by the stakeholders and only after that the different teams start working on their tasks (Larman, 2008).

Next comes a Design phase where the software to implement is designed and documented. This design documentation is delivered to the Implementation team where developers start coding.

Once the product coding is completed it is delivered to the Testing team who performs extensive code testing before it reaches the desired quality and is released to the customer.

After the software product is delivered to a customer, a Maintenance phase starts where fault corrections and upgrades are produced and delivered.

Throughout the process, there are strict quality milestones that have to be reached before jumping to the next process step. Also, the process is based on building documentation and producing deliverables to the next phases.

This approach has strengths and weaknesses (Larman, 2008):

- **Strengths:**

  On the strength side is the fact that the process is very logical, disciplined and structured. It is easily understandable, explainable and provides tangible clear milestones in the development process. The effort placed in the beginning, thinking and predicting as much as possible the product requirements and design, can lead to great effort economy in later stages. Also, the Waterfall life cycle is very valuable for its emphasis on documentation mind-set in order to transfer knowledge between team members and to new team members.

- **Weaknesses:**

  The Waterfall life cycle drawback starts with the fact that usually it is not possible to predict everything in the beginning and especially much in advance. The product requirements may also not be all clear in the beginning and there may happen unpredicted constrains throughout the production that may compromise the initial plan. Also, throughout the process may appear new good ideas to enhance the product

that may not easily and quickly be implemented because it was not planed and designed in the beginning.

The written documentation is not a proof that the knowledge can be efficiently spread because the document can simply not be read, misunderstandings and misinterpretations can happen when the document is read. Also, because it is not possible to write everything that is in the writer's mind. People are involved in the process, so there are team borders and work handovers between phases that can lead to adversarial relationships instead of strengthen team work to achieve a common goal. Therefore, the overall picture of waterfall life cycle weaknesses is that it is a rigid and change-resistant process that may lead to poor quality product if there are significant drawbacks and changes in requirements during the process are significant.

2.2 - Agile Development

Agile development methods have their roots in the old Iterative and Incremental lifecycle approaches.

They were developed with the belief on an approach more centred in human reality instead of process reality. Agile development focus on cross-functional teams empowered to make decisions, in opposition to big hierarchies and teams divided by functions. Agile pretends to emphasize building working software that can quickly be used, in contrast with spending long time in writing specifications. Also, Agile focus on rapid iteration with continuous costumer input during the development process, instead of defining all requirements in the beginning and only deliver the product to customer at the end (Larman, 2008).

2.2.1 - Iterative Development

This is a development method in which the overall lifecycle is composed of a sequence of iterations. Each iteration is a mini-project composed of requirement analysis, design, implementation and test activities. The challenge and goal of each iteration is to generate an iteration release which is a stable, integrated and tested part of the complete product. So, there will be several internal releases, at the end of each iteration, and one final release with the complete product in the last iteration of the process. This final release is then a product that can be released to the market or to a customer.

An iteration can be only focused on fault resolution or performance tuning, but usually there is an addition of new functionalities in each iteration. Therefore, the product incrementally increases iteration by iteration. Joining together the concept of developing a product iteratively and incrementally, there it is got the so called Iterative and Incremental Development, or simply Iterative Development (Larman, 2003).

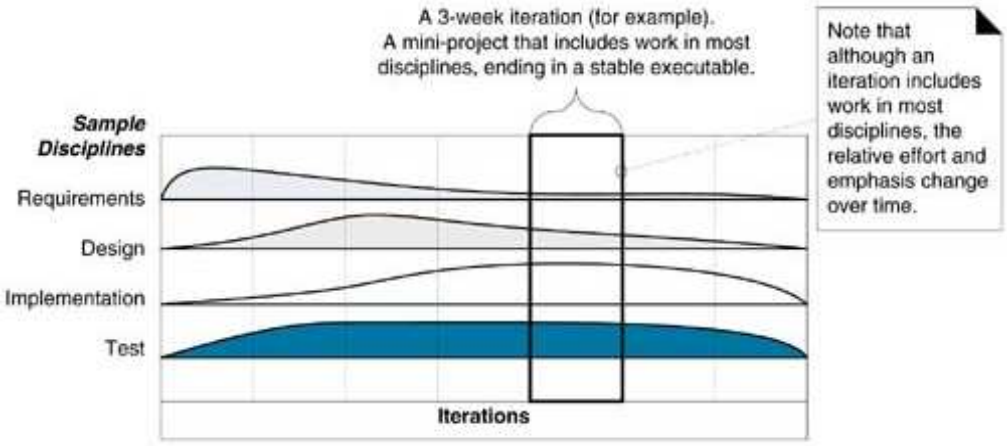


Figure 4 - Disciplines effort distribution across iterations (Larman, 2003)

2.2.2 - Agile Manifesto

In 2001, the Agile Alliance [1] was created by a group of people interested in Iterative and Agile methods. They defined the Agile manifesto and statement of principles [2] which are presented below (Larman, 2003):

Agile Manifesto Values

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

Table 1 - The Agile Manifesto Values (Larman, 2003)

[1] *www.agilealliance.com*
[2] *agilemanifesto.org*

The 12 Agile Software Principles

1. Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.	2. Working software is the primary measure of progress.
3. Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.	4. Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
5. Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.	6. Continuous attention to technical excellence and good design enhances agility.
7. Business people and developers must work together daily throughout the project.	8. Simplicity--the art of maximizing the amount of work not done--is essential.
9. Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.	10. The best architectures, requirements, and designs emerge from self-organizing teams.
11. The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.	12. At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behaviour accordingly.

Table 2 - The 12 Agile Principles (Larman, 2003)

2.2.3 - Agile Methods

Several Agile software development methods exist. Examples of these methods are:

- Scrum;
- eXtreme Programming (XP);
- Crystal;
- Feature Driven Development;
- Rational Unified Process;
- Adaptive Software Development.

eXtreme Programming (XP) and Scrum are the most used. eXtreme Programming primarily focuses on development practices (like pair-programming) and Scrum focuses on project management (Hossain, 2009a).

Since Scrum was the method used in the project under analysis, the next section will provide a Scrum method overview.

2.2.4 - Scrum Overview

Scrum is the most used and spread Agile method (Larman, 2008). It is an Iterative and Incremental framework for the development of products or applications that emphasizes a set of Project Management values and practices.

A common characteristic in Scrum is the timeboxing practice, i.e., limiting activities to a pre-defined amount of time. Timeboxing is applied in several meetings (events) defined in Scrum. A good example is the Daily Scrum meeting, which as a maximum duration of fifteen minutes in order to keep people focused on the meeting's goal and not generate additional discussions. Timeboxing is also applied in Scrum by limiting iteration's duration. Iterations in Scrum are called Sprints and take from one to four weeks. They are always of fixed duration and therefore, they end on a specific date whether or not the Sprint work is complete.

The Sprint starts with a cross-functional team meeting, selecting customer's requirements from a prioritized list and committing to complete those selected items until the end of the Sprint. These selected items never change during a Sprint. Every day each Team gets together in the so called Daily Scrum meeting. There, they present to each other the progress and update simple charts that orient them to the remaining work (Larman, 2008).

At the end of the Sprint, the Team reviews the Sprint with stakeholders and performs a Demo of what they have built. The feedback received can be incorporated in the next Sprint (Larman, 2008).

Scrum emphasizes the mean of getting a working product at the end of each Sprint. In software engineering this means an integrated and tested code, potentially shippable to the customer (Larman, 2008).

Development involves learning, innovation and challenges. Taking this in mind, Scrum is driven by the "Inspect and Adapt" mind-set. Therefore, short steps of development are followed by inspecting both the product developed and current practices' efficiency and by, adapting the product goals and process practices. Scrum emphasizes empirical rather than defined processes (Larman, 2008).

The figure below summarizes very clearly the Scrum Practices, Roles and Work products:

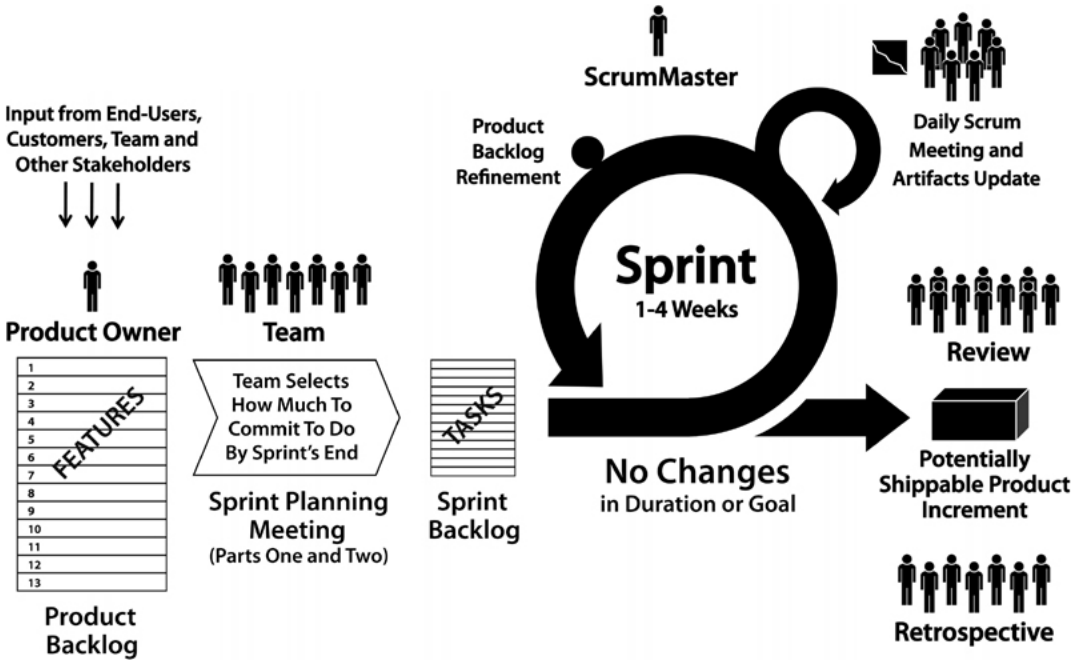


Figure 5 - The Scrum roles, work products and events (Larman, 2008)

Scrum Roles

In Scrum there are mainly three roles: Product Owner, the Team and the Scrum Master.

- **Product Owner** (Larman, 2008):

  His main role is to identify product features, translate these into a prioritized feature list, select features for the next Sprint and continually re-prioritize and redefine the feature list. He is responsible for the Return on Investment (ROI) and for the profit and loss of the product. For an internal application, typically the Product Owner and Customer can be the same person.

  In Scrum there is one and only one person who has the final authority of Product Owner, although it may exist other people serving as Product Owner, like a Proxy Product Owner.

- **The Team** (Larman, 2008):

  This is the group of people who effectively implement the product and provide ideas to the Product Owner on how to improve it. The Scrum Team is a cross-functional group (analysts, developers, designers, testers) who have the necessary expertise and are committed to deliver a potentially shippable product in each Sprint. It is also self-organizing and each member has a high degree of autonomy and

accountability and is not led by a manager or a project manager. The Scrum Team decides what they will commit in each Sprint and the best way to fulfil that commitment. The Team should be fully dedicated to the work of a Sprint, avoiding multitasking between different products or projects. In application groups with many people, it is possible to be organized into multiple Scrum Teams each one focused in specific features of the product. In this case, Teams are also known as *Feature Teams*.

- **Scrum Master** (Larman, 2008):

The Scrum Master is the Scrum mentor and coach to the Team. He is neither a manager nor a project manager and therefore, does not assign tasks or tell people what to do. On the other hand, the Scrum Master serves the Team, ensures Scrum practices are applied, protects the Team from outside interference and coaches both the Product Owner and the Scrum Team in the skilful use of Scrum. Scrum Master does everything what is in his competence to help the Team to be successful. He ensures that everyone understands and follows the Scrum practices and helps driving the organization through the changes required to achieve success with Scrum.

Each Scrum Team should have a fully dedicated Scrum Master. Although, it is possible in smaller Teams that one of the team members plays also the role of Scrum Master together with his role in the Team. Scrum Master and Product Owner can never be the same person.

Scrum Events

- **Starting Scrum** (Larman, 2008):

The first step in Scrum is an event where the Product Owner builds up the product vision and creates a list of features where all stakeholders can contribute. This list of features is known in Scrum as **Product Backlog (PBL)**. It may also include use cases, engineering enhancements, defects and each item has a defined priority. Only one Product Backlog exists in the product development and it evolves over the lifetime of the product with new items (new ideas, moves for competition, changes in customer needs) and re-prioritization decisions. Items (requirements) in the Product Backlog are also known in Scrum as **User Stories (US)**.

A subset of the Product Backlog is created containing features defined for the next product release. This subset of the Product Backlog is known as **Release Backlog**.

The Team provides to the Product Owner the effort estimate required for each item in the Release Backlog and the Product Owner assigns a business value estimate also to each item. With these two estimated values, and optionally with a risk estimate, the Product Manager prioritizes the Release Backlog to maximize the ROI or to reduce some major risk.

In Scrum it is also a common practice the use of relative estimates expressed in **Story Points (SP)** instead of absolute units, like person-hours. Although this practice is highly used in Scrum, it is not mandatory. SP is an arbitrary measurement used to measure the effort required to implement a User Story. Teams need to define, in the beginning of the project, a baseline User Story which all of them can relate to. From then on, all estimation should be done compared to that baseline User Story. As this measure is intrinsic to Teams in a project, it can never be compared between different projects.

- **Sprint Planning** (Larman, 2008):

The Sprint Planning Meeting is a meeting that takes place in the beginning of each Sprint and is divided in two distinct parts:

1. Sprint Planning Part One (What)

The aim of part one is to understand What the Product Owner wants! The Product Owner and the Team, with the support from Scrum Master, go through high-priority items in the Product Backlog, that the Product Owner is interested to implement in that Sprint. The goals and context for the high-priority items are discussed, providing the Team a better idea what Product Owner thinks.

Also, in this first meeting, Product Owner and the Team review the Definition of Done (DoD) that all items must meet in the end of the Sprint.

2. Sprint Planning Part Two (How)

This second part if focused in the detail task planning for How to implement the items that the Team decided to commit. The Scrum rules do not require the Product Owner to be present in the second part of the meeting, but he must be available, for instance by phone.

The Team selects items for the Product Backlog, starting from the highest priority, which they commit to finish until the end of the Sprint. This is a key practice

in Scrum and is one of its key characteristics: the Team decides how much work they will commit to complete in the Sprint, instead of having the work assigned to them by the Product Owner.

Due to the importance of the commitment, detailed analysis is needed to achieve success and so this meeting can take several hours.

The Team usually starts part two by estimating how much time each member has for Sprint related work (effective work without meeting participation, emailing, lunch breaks,…). Then, the Team starts with the first item on the Product Backlog, i.e., with the highest priority item for the Product Owner, and working together break it down into individual tasks and record them in the so called **Sprint Backlog**. This process repeats down in the Product Backlog item until all effective Team working hours in the Sprint are filled. At the end of the Sprint Planning meeting the Team has produced a list of all the tasks, with effort estimates, that they commit for the Sprint and the new Sprint Backlog is finished.

- **Daily Scrum (Larman, 2008):**

This is a daily meeting attended by the Team and occurring every workday in a Sprint. It lasts a maximum of fifteen minutes and the Team remains standing to keep the meeting brief. In this meeting the Team report to each other the progress and obstacles. Each Team member answers to these three questions:

1. What have you done since the last Scrum?
2. What will you do between now and the next Scrum?
3. What is getting in the way (blocks) of meeting the iteration goals?

There is no discussion. If discussion in needed it takes place afterwards immediately next to the Daily Scrum in the follow-up meeting. Someone in the Team takes notes of the blocking points and the Scrum Master is responsible to help the Team to solve and overcome the blocking points.

The Daily Scrum is not a status meeting to a manager. Instead it is the place for the self-organizing Team to share openly with each other what and how is going on and to help them to coordinate and keep synchronized.

Scrum does not recommend having managers or others who perceived authority in the Daily Scrum. Team can feel monitored and pressured to report major progress and inhibited them to openly report problems. This also can have a negative impact in the Team’s self-management.

- **Updating Sprint Backlog (Larman, 2008):**

The Team updates everyday their estimate of the amount of time remaining to finish their current task in the Sprint Backlog. After this update, is added up the hours remaining for the Team as a whole and a chart is plotted. This chart is known as **Sprint Burndown Chart**. It shows every day the new estimate of how much work in hours remains until the Team's tasks are finished. Therefore, it shows the Team's progress towards their goal in the end of the Sprint.

It is desired that the plot has a downward trend, with a trajectory to reach zero effort remaining in the last day of the Sprint. If the Team is deviating much from their goal they have to adjust, either reducing the scope or finding ways to increase the efficiency, but maintaining a sustainable pace.

- **Product Backlog Refinement (Larman, 2008):**

This is a practice that typically takes 5% to 10% of the Team's work in each Sprint. The Product Backlog Refinement includes tasks like detailed requirement analysis, splitting large items into smaller ones, estimation of new items and re-estimation of existing items. There are no rules from Scrum how the refinement work is done. But it is recommended to be a focused workshop near the end of the Sprint where the Product Owner and the Team are fully focused on these tasks. With this, Sprint Planning can be simpler. This because Product Owner and the Team start the Sprint planning with a clear, analysed and carefully estimated items.

- **Sprint Review (Larman, 2008):**

Happens after the end of each Sprint and is where the Team reviews the Sprint with the Product Owner.

This is an **Inspect and Adapt activity regarding the Product**. The Product Owner learns what is going on with the Product and the Team with the Product Owner and the market. Therefore, the most important element of the Sprint Review is the in-depth conversation between the Team and the Product Owner.

The Sprint Review includes a Demo of what the Team has built during the Sprint. The Demo is not a slide ware presentation, but a real demonstration of the working software of the Sprint's build.

The Scrum Master is responsible for knowing the Definition of Done (DoD) that was defined in the Sprint Planning. During the meeting he is responsible to tell the Product Owner if items implemented by the Team did not meet the DoD. This practice increases the visibility of the work quality and prevents the Team to fake the quality of the build.

In this Review meeting are present the Product Owner, Team members and Scrum Master. In addition, customers, stakeholders, experts, executives and anyone else interested are also present.

- **Sprint Retrospective (Larman, 2008):**

This is a meeting that happens after the Sprint Review and involves **Inspect and Adapt regarding the Process**. It is the opportunity for the Team to discuss what is working and what is not working and agree on changes to try.

In this meeting, the Team and the Scrum Master are involved. The Product Owner is welcome, but not required.

The Scrum Master can act as a facilitator for the retrospective, but is recommended to keep a neutral position to facilitate the meeting. A good approach is the Scrum Master facilitates the Scrum Team of other's Scrum Master on the Retrospective.

Team members fill what they think in the columns "What's working well" and "What could work better" of a whiteboard. Repeated items are marked to be clear at the end what the common reported items are. Afterwards the Team looks to the related causes and agrees on a small number of changes to be tried out in the upcoming Sprint and reviewed in the next Retrospective.

- **Updating Release Backlog and Release Burndown Chart (Larman, 2008):**

After the end of each Sprint, some items in the Release Backlog have been finished, some have been added, some have new estimates, and some have been dropped from the release goal. Product Owner is responsible to keep these changes updated in the Release Backlog.

Additionally, there is the Release Burndown Chart that shows the current progress towards the release date. This is similar to the Scrum Burndown Chart, but is related with the Product Release.

Scrum Values

- **Commitment**

"The Scrum Team commits to a defined goal for an iteration, and is given the authority and autonomy to decide themselves how best to meet it. Management and the Scrum Master commits to not introduce new work during an iteration, avoid directing the Team, and work to provide the resources and quickly remove blocks that the Team reports in their daily Scrum meeting. The Product Owner commits to define and prioritize the Product Backlog, guide choice of the next iteration's goals, and review and provide feedback on the result of each iteration." (Larman, 2003, p. 246)

- **Focus**

"The Scrum Team has to be able to focus on the stated goals of the iteration, without distraction. Thus, management and the Scrum Master focus on providing the Team with resources, removing blocks, and avoiding interrupting the Team with additional work requests." (Larman, 2003, p. 246)

- **Openness**

"The openly accessible Product Backlog makes visible the work and priorities. The Daily Scrums make visible the overall and individual status and commitments. Work trend and velocity are made visible with the Backlog Chart." (Larman, 2003, p. 246)

Respect:

"Or, Team responsibility rather than scapegoating. The individual members on a Team are respected for their different strengths and weaknesses, and not singled out for iteration failures. The whole Team rather than a manager, through self-organization and direction, adopts the attitude of solving "individual" problems through group exploration of solutions, and is given the authority and resources to react to challenges, such as hiring a specialist consultant to compensate for missing expertise." (Larman, 2003, p. 246)

- **Courage**

"Management has the courage to plan and guide adaptively and to trust individuals and the Team by avoiding telling them how to get the iteration done. The Team has the courage to take responsibility for self-direction and self-management." (Larman, 2003, p. 246)

Scrum Software Metrics

Measuring is an essential practice to effectively manage and control any activity. Without exception, it is a common practice in software engineering where software metrics are defined in order to assess process efficiency and developed product quality. In other words, the goals of software metrics are the identification and measurement of the essential parameters that affect the software development (Mills, 1988).

Measurement is also included in industry standards like CMMI from Software Engineering Institute (SEI) (Westfall, 2005). CMMI-DEV does not specify that a project or organization must use some specific set of metrics and that need to achieve some specific performance target. Although, it defines that a project or organization should have process for development practices and define sets of Generic Goals and Generic Practices for each of the model's maturity levels. Measurement is specifically addressed in CMMI maturity level 4, named "Quantitatively Managed", where is defined that an organization and projects need to establish quantitative objectives related to quality and process performance. In order to obtain quantitative information as management criteria, metrics to measure process performance and product quality need to be defined. These metrics and quantitative information are essential for an organization in order to evolve to maturity level 5 where Optimization goals are addressed (CMMI-DEV v1.3).

In software engineering many sets of product metrics are used and typically measures product's size, complexity and quality. Traditional software development typically measures product size with the number of Lines of Code (LOC) or the number of Functional Points. Developed product's quality is typically measured by means of defect metrics, where are considered the number of faults detected during testing phases. (Mills, 1988)

Scrum development continuously measures both the product and the process to develop it (Hartmann, 2006). Typical metrics in Scrum development are:

- **Burn Up** - measuring product's size and the velocity of the development;
- **Burn Down** - measuring remaining work until the end of a release/product;
- **Build status** - measuring usefulness of produced builds,
- **Opened Faults** - measuring overall product's quality.

In order to measure Teams' accomplishment ratio in each Sprint it is used the **Descoped**[3] metric. It measures the amount of work that was estimated and planed in the beginning of each Sprint and what was effectively achieved at the end of that Sprint.

Dubinsky et al. (Dubinsky, 2005) used four Agile metrics in their study in the Israeli Air Force. Those metrics presented information about the amount and quality of work that was performed, about the pace of the work progress, and about the status of the remaining work versus the remaining human resources. Table 4 provides a summary with the used Agile metrics:

Metric	What is measured	How is measured
Product Size	The amount of completed work	The number of test points
Pulse	The continuous integration	The number of check-ins per day (code, automatic-tests and detailed specifications)
Burn-Down	The project remaining work versus the remaining human-resources	The number of story point still to be developed and remaining available human resource effort.
Faults	The product's quality	The number of faults per iteration

Table 3 - Agile metrics used in Israeli Air Force project case study (Dubinsky, 2005).

[3] There were not found literature references about this Scrum metric, but it was used in the project in study to measure team's accomplishment ration in each Sprint.

2.3 - Global Software Development

In 2001 Herbsleb et al. (2001) stated in their article about Global Software Development (GSD) that a steady and irreversible trend towards the globalization of business, and in particular the software business, have been happening since previous decades.

Markets are moving from national to global causing a profound impact on how products are conceived, designed, constructed, tested and delivered to customers (Herbsleb, 2001).

This economic pressure, together with an extreme competitive business environment, forced organizations to manage new ways of improving effectiveness and efficiency as well as increasing customer satisfaction.

These drivers lead software companies to begin experiments with remotely located software development and with outsourcing. Benefits related to Global Software Development (Herbsleb, 2001) are:

- Reduce development costs by getting low labour costs in some parts of the world;
- Improve time-to-market by taking benefit of the time zone differences and “round-the-clock” development;
- Access to a global pool of resources wherever located;
- Proximity to the market, knowledge of customers and local conditions;
- Quick formation of virtual corporations and virtual teams to exploit market opportunities.

Global Software Development is characterized by distributed teams composed by stakeholders from different national and organizational cultures, different geographical locations and typically different time-zones (Jalali, 2010). This environment face engineers, managers and executives with huge challenges on technical, social and cultural levels (Herbsleb, 2001) and have significant effects on communication, coordination and control (Jalali, 2010). There is evidence that multisite development tasks take much longer when compared with collocated tasks and that this delay is directly related to communication and coordination roles (Herbsleb, 2001).

Herbsleb et al. (2001) have listed the problems and challenges originated by physical separation and distance in GSD:

- Strategic issues;
- Cultural issues;
- Inadequate Communication;
- Knowledge management;
- Project and Process management issues;
- Technical issues.

Paassivaara et al. (2009) performed a literature review in 2009 on the supporting GSD practices to provide solutions to the raised challenges. Their findings are summed up in the following table:

Practice Name	Description
Frequent visits	Used to build and maintain trust and enhance collaboration. Seeding visits early in the project aim at building a relationship. Maintaining visits are shorter and aim at maintaining the collaboration. Team members should continually rotate between sites and at least one team member at a time should be visiting.
Multiple communication modes	Several different kind of communication should be available that can also be applied in parallel, i.e., individual and conference telephone, teleconference, videoconference, email, instant messaging, Wiki and desktop sharing
Mirroring/ balanced sites	Reduce the dependency between sites. Each role in a team at one site has a counterpart on the other site.
Ambassador/ rotating guru	Experienced engineers are sent to the other site for a longer period of time. Ambassadors report lessons learned and set future directions for the projects. Rotating gurus provide initial training and mentoring to the other site.
Synchronization of work hours	Maximization of overlapping work hours to ensure constant communication. For example, early morning shifts for one site and late evening shifts for the other site.

Table 4 – Supporting GSD Practices (Paasivaara, 2009)

Some of the different terms used in the GSD are next described (Jalali, 2010):

- **Outsourcing** (offshore/onshore outsourcing): an external company is responsible for providing software development services or products to a client company. If the subcontracting and client companies are located in the same country, this is known as onshore outsourcing. If they are located in different countries is known as offshore outsourcing.
- **Offshoring** (offshore insourcing): a company created its own software development centres distributed by different countries to handle their production.
- **Nearshoring:** similar to Offshoring, but the companies distribute their development centres over nearby countries normally sharing a border.
- **Distributed team**: team members are distributed over different locations and work remotely on different parts of a project, with or without face-to-face interactions.
- **Co-Located team**: A team together in the same team room. A team working in the same city is not classified as "co-located" (Larman, 2010)

2.4 - Distributed Agile Development

Agile Development gained recently interest due to its flexible approach managing requirement's volatility, promotion of a close collaboration between customers and development team, and early and frequent delivery of product releases (Hossain, 2009a).

There is also a growing interest in applying Agile methods in Global Software Development to boost the advantages of both approaches (Hossain, 2009a).

But, as seen in the previous topic, distributed development raises huge challenges regarding communication, teamness, cultural differences and time-zone differences. How can Agile methods be applied in distributed development projects being Agile based on open, constant and face-to-face communication, on an informal process and self-organized teams? How can Agile principles like:

- Business people and developers must work together daily throughout the project;
- The most efficient and effective method of conveying information to and within a development team is face-to-face conversation;
- Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done;

be applied and followed in distributed development?

Ramesh et al. (2006) performed an analysis about theoretical characteristics of Distributed and Agile Development. They provided an overview of differences and potential conflicts between both development practices, raising questions on potential challenges to be faced Author's analysis is next presented in Table 5.

Distributed Development	Agile Development
Communication need vs Communication impedance [4]	
Distributed development is based on formal mechanisms like detailed architectural design and plans addressing impediments to team communication due to geographical separation.	Agile development is based more on informal interactions than explicit documentation.
"How can we achieve balance in formality of communication in Agile distributed environments?"	
Fixed vs Evolving quality requirements	
Distributed development typically relies on fixed, upfront commitments on quality requirements.	On the other way around, Agile development relies on ongoing negotiations between development team and the customer for determining the acceptable levels of quality at various stages of development.
"How can we achieve a balance between the fixed and evolving quality requirements?"	
People vs Process-oriented control	
In Distributed development the control is normally achieved by defining formal processes.	On the other hand, Agile development is more people-oriented and the control is established in a more informal process.
"What would be the appropriate balance between people-oriented and process-oriented control in Agile distributed development?"	
Formal vs Informal agreement	
Distributed development relies on explicit targets, milestones and detailed specification of requirements.	In Agile development the contracts are loosely and informally defined.
"What is the appropriate level of formality in developing contractual agreements in Agile distributed development?"	
Lack of team cohesion	
In Distributed development with the distribution over different sites participants are less likely to perceive themselves as part of the same team when compared with co-located participants. This generates lack of cohesiveness and shared view of the goals.	In Agile development these problems are even more pronounced as these methods emphasizes constant cooperation on all aspects of the project.
"How can team cohesion be improved given the constrains of a distributed environment?"	

Table 5 – Distributed Development vs Agile Development (Ramesh, 2006)

[4] Communication Impedance expression is used to refer the communication resistance in dynamic environments.

2.4.1 - Hossain and Jalali Research Studies'

With the growing interest in Agile methodologies and GSD, a significant number of studies in this area were published (Jalali et al. (2010) refer to 77 and Hossain et al. (2009a) refer to 20) reporting real-life empirical case studies and industry reports.

The information is not available under a consistent and systematic way. In the recent years there have been efforts with systematic literature reviews to summarize and consolidate the information published in the last ten years. Two examples are the systematic literature reviews from Jalali et al. (2010) and Hossain et al. (2009a) on the theme of the applicability of Agile methods in GSD projects. Both reviews followed the same guidelines to conduct a systematic literature review defined by Kitchenham and Charters (2007). Jalali et al. (2010) review was conducted for all Agile methods applied in GSD while Hossain et al. (2009a) focused their review specifically on Scrum method in GSD.

Jalali et al. (2010) systematic literature review aimed to summarize the existing literature and to investigate which Agile practices have been used in a GSD context. Authors have provided the scientific community with an overview of the status in the area, highlighting the gaps. Their review tried to answer these two questions:

1. *What is reported in the current peer-reviewed research literature about the Agile practices in GSD?*
2. *Which Agile practices, in which GSD settings, under which circumstances have been successfully applied?*

Hossain et al. (2009a) review goal was to explore, investigate and explain various challenging factors as well as current strategies to address those challenging factors. Authors restricted the study to the use of Scrum in GSD. The review tried to answer the broader question:

1. *What is currently known about the use of the Scrum practices in GSD projects?*

and more specifically to the following two questions:

2. *What challenging or risk factors restrict the use of Scrum practices in globally distributed projects?*
3. *What strategies or practices are being commonly used to deal with these challenging factors to support the use of Scrum practices in globally distributed projects?*

Next will be presented and analysed the conclusions reached by Jalali et al. (2010) and Hossain et al. (2009a) in their reviews.

Growing Interest

The applicability of Agile methods in Global Software Development is not yet well investigated (Jalali, 2010) and is still an open debate whether or not they can be successfully applied in distributed environment (Hossain, 2009a).

The interest in this theme has been increasing over the recent years. Hossain et al. (2009a) in their literature review have found an increasing number of articles from 2005 until 2008 (*Hossain article was written in 2009*), summarized in Table 6.

Year	2003	2004	2005	2006	2007	2008	2009
papers	1	1	1	3	4	9	1
%	5%	5%	5%	15%	20%	45%	5%

Table 6 – Hossain's selected papers about Agile in GSD by year (Hossain, 2009a)

Jalali et al. (2010) also came to a similar conclusion about this growing interest. The authors have described in their study the number of articles related with the theme coming from different sources, as presented in Table 7. It can also be seen that 2008 was the year with most published articles and that there was a significant increase during the last five years.

	1999	**2000**	**2001**	**2002**	**2003**	**2004**	**2005**	**2006**	**2007**	**2008**	**2009**
ACM						2	2	3			2
IEEE					1	2	1	2	6	15	9
Compendex				1		1	2	4	4	2	2
Inspec					1	5	1	3	2	1	
AIS									1	2	
Total	**0**	**0**	**0**	**1**	**2**	**10**	**6**	**12**	**13**	**20**	**13**

Table 7 – Jalali selected papers about Agile in GSD by year and source (Jalali, 2010)

It was also verified in both studies that the interest in Agile practices in GSD started around 2002/2003.

Research type

From the articles analysed by Hossain et al. (2009a) it was found that only 20% of them were empirical studies. The rest of the 80% were classified as "lessons learned" or industrial experience reports. Thus, Hossain et al. (2009a) concluded that there is little evidence based on empirical reports about the use of Scrum in GSD.

In Jalali et al. (2010) study was verified that the majority of the studies in literature were in the form of experience reports (Figure 6), i.e. on the personal experience of practitioners about a particular issue and the method to address it. Although experience reports are valuable, the authors concluded that there is lack of evaluation research where more rigorous research methods and literature reviews are required. It was proposed a close collaboration between academia and industry where the research part is done on academia and data collection done from real industrial cases. Jalali et al. (2010) recommend future research in this theme to follow the guidelines presented by Peterson and Wohlin (2009) to structure the context for empirical industrial studies.

In contrast with Hossain et al. (2009a) study, Jalali et al. (2010) have analysed 77 articles, being 60 of them (78%) empirical studies (Figure 8). This can be an indication that in the broader theme of applying Agile in GSD there is more empirical evidence than in the more specific theme of applying Scrum in GSD.

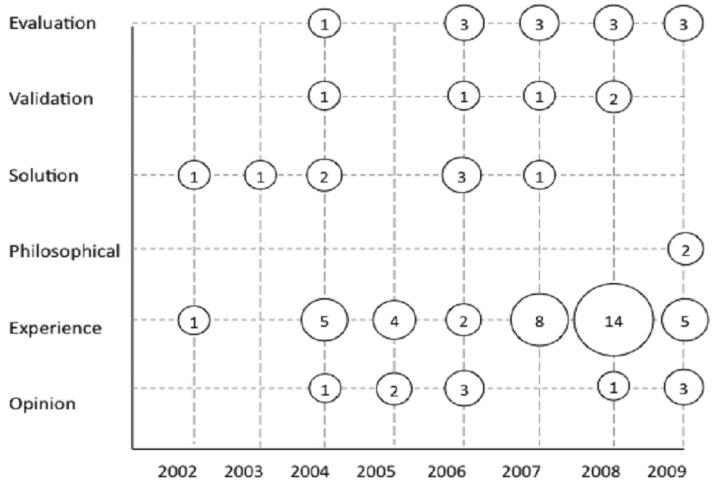


Figure 6 - Jalali distribution of research types over the studied years (Jalali, 2010)

Project Characterization

Hossain et al. (2009a) verified that most of the articles reported projects using Scrum practices in distributed development environment on intra-organizational multinational companies. Projects used mainly two sites, with more than two Scrum Teams and more than twenty five persons per Team. Most of the projects faced the challenge of not having overlap working hours, although several reported cases have some overlap working hours. The majority of the project were of Web development, although it was also verified a significant diversity on other areas.

Table 8 shows the detailed numbers about project categorization presented by Hossain et al. (2009a) in their literature review:

Collaboration Mode	Intra Organizational	16
	Inter Organizational	4
Number of Sites	Two	15
	Three	5
	Three+	0
Number of Teams	Two	6
	Two+	12
	Unclear	2
Team Size	Up to Twenty Five	5
	Twenty Five+	10
	Unclear	5
Time differences	No overlap time	10
	Overlap time	8
	Unclear	2
Application domain	Web	7
	Oil and Energy	1
	Library	1
	Logistic	2
	Public safety	1
	Airline	1
	Commercial	2
	Business service	1
	Finance	1
	unclear	3

Table 8 – Hossain studied distributed Scrum project categorization (Hossain, 2009a)

In Jalali et al. (2010) review was verified that most of the studies claimed applying "Agile" in projects as a general term, not always specifying exactly which Agile method was used. From the ones that have specified, Scrum and XP were the most common used ones. Regarding the team/organization setting in GSD was verified that "distributed team" was the most common, followed by the "Offshore" setting. But, some of the studies did not specify any organization setting. These findings from Jalali et al. (2010) revealed incomplete contextual and background information making it difficult to achieve solid conclusions from the studies in the literature. Figure 7 presents Jalali et al. (2010) findings on mapping of Agile and distribution type.

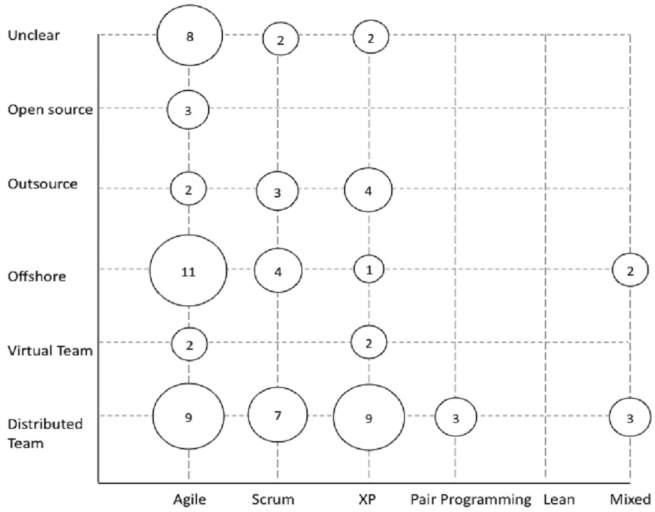


Figure 7 - Jalali mapping of Agile practices and distribution types (Jalali, 2010)

From the included studies by Jalali et al. (2010) 78% were empirical studies, where 63% of them where written by industry practitioners, 32% by academic researchers and 5% by joint industry practitioners and academic researchers. 82,5% of the empirical cases reviewed claimed success in applying Agile practices in GSD. Figure 8 presents a graphical representation of the number of successful and failed projects according to the research method.

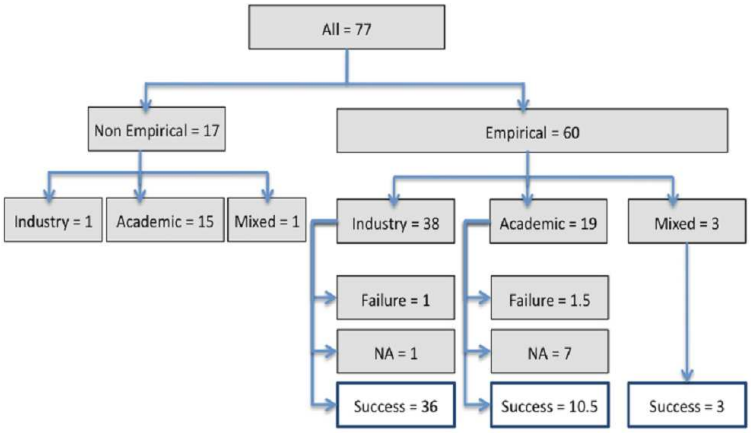


Figure 8 - Jalali number of success empirical studies on Agile in GSD (Jalali, 2010)

From the successful case studies, Jalali et al. (2010) identified the applied practices and their frequencies, summarizing this information in the following figure:

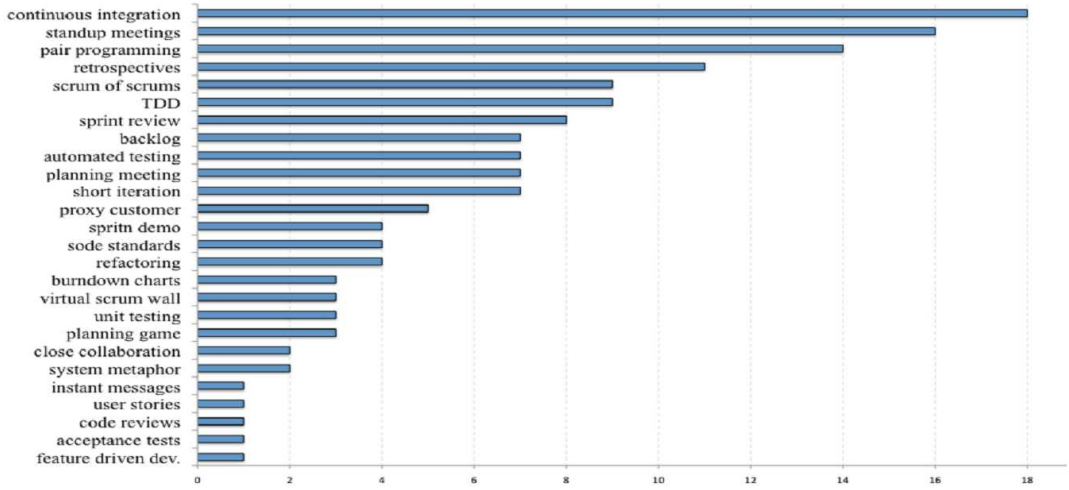


Figure 9 – Jalali Agile practices in reviewed studies on Agile in GSD (Jalali, 2010)

It can be seen from Figure 9 that continuous integration, daily stand-up scrum meetings, pair programming and retrospective are the most frequent practices. Jalali et al. (2010) verified that several practices were reported in the literature, but in many cases it was unclear which Agile method has been used or, on the other way around, some claimed to be Agile,

but few practices were referred to be used. Therefore, also here authors have doubts about the reliability of the reported results.

The success cases were also analysed to get information about the main project characteristics like size, duration, domain and knowledge area. As can be verified in Table 9 the most common project characteristics were distributed team, in long duration projects, on a small or medium size project. Also here is visible the lack of documented information about the context and project/organization settings resulting in a significant large number of unclear cases.

Distribution		**Global?**	
Distributed team	|||||||||||||||||||||||	Yes	||||||||||||||||||||||||||||||
Offshore	||||||||||||	No	|:
Outsource	||||||	Unclear	||||||||||||:
Virtual team				
Open source				
Unclear	|||			
Duration		**Size**	
Long	|||||||||	Large	|||||
Medium	|||||	Medium	||||||
Short		Small	|||||||||:
Unclear	||||||||||||||||||||||||||||||	Unclear	||||||||||||||||:
Knowledge Area		**Domain**	
Requirement	||:	Web	|||||||
Design	||	Automotive		
Construction	|||||	Service		
Testing	|||	Embedded		
SE Management	||||||	Telecom	|||
SE Process	||||	Real time		
Maintenance	||||	Commercial		
Tools & methods	||||	Business critical		
Unclear	||||||||||||||||	Finance		
		Unclear	||||||||||||||||||||||||||||||

Table 9 – Jalali summary of reviewed project's characteristics (Jalali, 2010)

Jalali et al. (2010) considered the classification of project size and duration as:

- Small <= 20 person < Medium <= 50 person < Large
- Short <= 1 month < Medium <= 7 months < Large

and the specification of the knowledge areas based on SWEBOK (Abran, 2004).

Limitations of the reported studies

Jalali et al. (2010) concluded that the contextual details of many of the reviewed empirical studies were insufficient, recommending researchers to design and use a template for documenting the contextual information which is not too detailed nor too abstract. It was also stated that industrial experience reports normally did not include related work and do not reference literature.

Hossain et al. (2009a) clearly came to the conclusion that there is the need to increase the quantity and the quality of empirical studies in the area in order to describe, evaluate, explore and explain the use of various Scrum practices in GSD practices.

Therefore, literature presented lack of details about projects and their contextual environment as well as about methods used, in order to achieve solid conclusions about the applicability of Agile methods in GSD.

Applicability of Agile in GSD

Hossain et al. (2009a) verified that most of the reviewed studies claimed some degree of success in applying Agile in GSD, but despite of these reports, the mechanisms of combining Scrum practices in GSD were not well understood. The success may be impacted by several contextual factors of the project. Studies have not considered the impact of factors like budget, complexity, criticality, team experience, and time constrains among others. Therefore, authors conclude that successful use of Scrum in GSD may be limited by several project's contextual factors.

Jalali et al. (2010) stated that there is no sufficient evidence to conclude that Agile is efficiently applicable in large distributed projects. Their conclusion refers that although there were some studies reporting experiences in large projects, not all the contextual projects were clearly reported.

Although the interest and the number of studies published about applying Agile in GSD has clearly increased in the last few years, there is still no clear evidence and solid conclusion about its complete success.

Challenges Faced

Jalali et al. (2010) verified that there are still not a sufficient number of studies analysing the challenges of applying Agile methods in GSD. Problems and challenges in GSD or in Agile are documented, but the combination of both is not enough examined in real world situations. Therefore, their conclusion was that there is the need for in-depth studying of challenges and benefits of combining Agile and GSD in the form of evaluation research.

Hossain et al. (2009a) verified that a number of challenging factors impact the GSD communication, coordination and collaboration process. These factors were caused by the temporal, geographical and socio-cultural differences due to the project stakeholders' distribution. Challenges related with communication were identified as vital and the cultural differences between the distributed team can have significant impact on the team's collaboration process. Managing a large Team distributed over different sites is quite challenging and the lack of tools and insufficient infrastructure support can be an obstacle to the Scrum practices in GSD.

Strategies and Extensions to the Agile practices

Jalali et al. (2010) refer in their conclusions that within many of the reviewed studies Agile practices have been customized and a modified. Authors strength the need for further investigation where modifications are well studied in order to provide guidelines on how to adapt the Agile practices. Additionally, it should be determined how much change is allowed so that it is still recognized as Agile practices in GSD.

Hossain et al. (2010) concluded that Scrum Teams need additional strategies suitable to their environment in order to support the use of Scum in GSD. Distributed Scrum Team may choose different Scrum Team models to reduce its project distribution challenges. In order to carry out several Scrum meeting practices some overlap time between distributed Teams is necessary.

If no overlap time is possible distributed Scrum Teams may use some practices like:

- synchronized work hours;
- local Scrum;
- additional local meetings;
- strict communication policy;
- key persons attending all distributed meetings;
- reducing number of Scrum meetings;
- asynchronous retrospective, among others.

Team collaboration can be enhanced by introduction additional practices like:

- team gathering;
- exchange visits;
- informal meeting of distributed Team members;
- mandatory presentations;
- maintaining key documentation; and
- gradual Team distribution.

Multiple modes of communication can also be applied in order to reduce the lack of communication bandwidth and tools. Distributed Scrum Team needs as well to be supported by several tools for project management, backlog management and tracking issues.

Hossain Conceptual Framework

Hossain et al. (2009b) went a step further and, based on their findings in the systematic literature review, proposed a conceptual framework addressing the key challenges and mitigation strategies when applying Scrum practices in GSD.

The conceptual framework categorizes the identified challenges in seven broad classes. For each of them, describes the current strategies and some practices reported in the literature to address those challenges. Authors intended with the framework to synthesize information and help GSD Project Managers to understand the key risks that may have negative impact when using Scrum practices.

The Hossain et al. (2009b) conceptual framework is presented in the below Table 10:

Challenges Classification	Current Strategies	Some Practices
Asynchronous	Synchronized Work Hours	- Adjust working hours between distributed sites to support distributed Scrum meetings
		- Allow distributed Scrum team members to attend meetings from home (e.g. by phone)
	Reducing Scrum Meetings Length	- Strict time boxed short meetings (e.g. thirty minutes planning meeting rather than longer)
		- Prior asynchronous work (e.g. backlog preparation) before attending meetings
	Site Based Local Scrum team	- Scrum of Scrums meeting practices attended by each Scrum team key touch points
		- Establish multiple communication channels (e.g. additional architectures Scrum of Scrums)
	Modified Scrum Practices	- Additional site based morning “mini Scrum” after late night distributed Scrum meetings
		- Only key members rather than whole site members attend in late distributed Scrum meeting
		- Reduce distributed meeting frequencies e.g. daily Scrum meeting twice in a week
		- Asynchronous Scrum meeting by posting meeting results on wiki or emailing meeting minutes
		- Sprint demo is conducted by only onshore team (closer to customer)
Lack of Group Awareness	Team Gathering	- Distributed Scrum teams perform few initial sprints in a single location as collocated teams
		- Scrum teams gather quarterly or annually for few days and perform some meetings
		- Gradual team distribution e.g. through evaluation, inception, transition, steady state stages
	Visit	- Product owners frequent offshore visits
		- Scrum management involvement in offshore kick off meeting
		- Planned rotations between distributed Scrum team members
	Additional Distributed Meeting	- Leadership meeting for example “unified planning meeting” attended by Scrum masters
		- Unofficial distributed Scrum teams QA or architectures meetings
		- Distributed Scrum team members socializing for example virtual party or internet games
	Training	- Initial Scrum training to reinforce the value of Scrum
		- “Technical Scrum” attended by distributed sites key members to clarify new technology issues
		- Product owners quarterly or even annually product road map meetings
	Key Documentation	- Maintaining valuable documents e.g. supplementing user stories with use cases in backlog
		- Extensive use of collaborative tools for example using Wiki to discuss development issues
	Mandatory Meeting Participation	- Distributed Scrum team members mandatory presentation in meetings (e.g. Scrum demo)
		- Distributed Scrum team members encouraged to provide additional information in meetings
Poor Communication Bandwidth	Multiple communication Modes	- Continuous network monitoring by a dedicated infrastructure team to ensure quality transmission
		- Ensure wide range of communication tools suitable to the network infrastructure
Lack of Tool Support	Proactive Resource Management	- Ensure sufficient skills and effective tools to support Scrum processes
		- Ensure wide range of collaborative tools e.g. enterprise wikis to support team collaboration
		- Ensure globally accessible product backlog, sprint backlog and burndown chart
Large Number of Project Personnel	Split Large Team	- Build autonomous sub teams that are capable of using Scrum processes
		- Each sub teams are allocated independent architectural subsystems
		- Sub teams based on feature
		- Sub teams based on function
Lack of Collaborative Office Environment	Single Room	- Ensure Scrum team members are located in a single room
		- Try to arrange split team members in a single room
		- Virtual single room (for example using dedicated team wiki) for a distributed Scrum team
	Dedicated Meeting Room	- Ensure separate meeting room for each distributed site with necessary network and tools
		- Try to make distributed team members visible in Scrum meetings e.g. use a video projector
		- Use virtual conference room for a distributed Scrum team
Increased Number of Sites	Site Based Scrum Team	- Form autonomous local Scrum team and allocate independent architectural sub systems
		- Scrum of Scrums practice for inter-site team communication
		- Scrum of Scrum of Scrums practice for a large number of Scrum teams involvement
	Restricted Scrum Team Distribution	- Maintain team distribution policy e.g. Scrum team will not be distributed more than two sites

Table 10 – Hossain conceptual framework (Hossain, 2009b)

As described above in the systematic literature review conclusions, there is low evidence in the literature about the challenges in applying Scrum in GSD. Also, there is lack of detail in the studied project's characteristics and context. Therefore, Hossain et al. (2009b) do not claim having developed an exhaustive list of components identifying all the challenges and corresponding strategies to address them. Other framework's limitation stated by authors was concerned with the fact that it was based on the review of only twenty studies identified in the systematic literature review.

As future work, Hossain et al. (2009b) plan to continuously modify the proposed framework based on literature findings as well as conducting multiple in-depth industry-based case studies in real life settings.

Therefore, the conceptual framework proposed by Hossain et al. (2009b) is currently not complete, but is a first approach to consolidate and synthesize the issues in applying Scrum practices in GSD and should be subject of future evolution with further investigation of real life industry project.

2.5 - Literature Review Conclusions

The interest in the theme of applying Agile practices in Global Software Development has been increasing over the recent years. The growing number of reported studies since 2002 and especially in the last five years is an evidence. But, although there is this growing share of knowledge, the information is spread and not provided in a systematic way.

There were efforts in the last two years to concentrate and consolidate the information available in the scientific community about the applicability of Agile practices in GSD with systematic literature reviews.

From these literature reviews it was seen that the majority of the articles in the literature are industrial experience reports and that there is lack of evidence based on empirical reports. It has been proposed more evaluation research where more rigorous research methods and literature reviews are required.

Within the articles in the scientific community there is a significant number reporting success in applying Agile methods in GSD. But it was also concluded that there is lack of details about the projects studied and their contextual environment which do not allow achieving solid conclusions about the successful applicability of Agile methods in GSD.

Although the information in the literature does not exhaustively identify the challenges in applying Agile practices in GSD and the strategies to address them, Hossain et al. (2009b)

have proposed a conceptual framework. There, authors categorize the identified challenges in seven broad classes and, for each of them, describe the current strategies and some practices reported in the literature to address the challenges.

Despite this conceptual framework is currently not complete, it is the first approach to consolidate and synthesize the information and would be subject of evolution in the future with more real life industry project investigation.

After this literature review, it can be concluded that there is still a lot of research to be performed in order to fully understand how Agile methods can be efficiently applied to Global Software Development. Regarding the theme's importance and its growing interest it was expected that scientific knowledge would be currently more advanced and consolidated. This dissertation may contribute for the improvement on this research area by providing a structured description of the case study context environment as well as to validate, and possible extend, the proposed conceptual framework by Hossain et al. (2009b) within the project reality in.

This page intentionally left blank

3 - Case Study

In this chapter the case study will be described and its results and conclusions presented.

This chapter starts with the case context description in order to frame the environment of the project. It continues with the description of the Scrum practices used in project together with the support and communication tools available. It will be also presented the development performance evolution along the project by analyzing several metrics collected during the study. In the section "Challenges and Impediment" semi-structured interviews' results will be presented and analyzed. This chapter is finalized with the overall case study conclusions.

Complementary and additional details about the study results can be found in Annex A to Annex E.

3.1 - Case Study Context

In this section, the studied project's context will be described.

Starting with a description of the organization where the project was being held, the section continues with the description of the product developed and its target market. A description of the project organization, together with its scaling along the project duration, is performed in order to provide a good support overview of the project evolution and the study context.

3.1.1 - Organization

The company where this case study was performed is a global corporation following a matrix-organization schema. The product was being developed under a Research & Development unit that included also Program Management and Product Management teams. As a global corporation, organizational units were distributed globally over different sites.

Agile methodologies, in particular Scrum, were gradually introduced in the company during the last four years. Until then, waterfall life cycle was mainly used in development units. An interim period was used to run pilot Scrum projects and to evolve projects from plan-driven/waterfall to Scrum (vision-driven/interactive). Today every new project is managed using Scrum methodologies from the beginning.

The project under study in this dissertation started in September 2009 and, from the very beginning, Scrum methodology was applied.

3.1.2 - Product

The software system developed by the object of study was an information system for network management developed under Java programming language. It was a market-driven product, i.e., with no customer identified at the beginning and trying to conquer the market.

The product was an evolution of a previous system version, but with a complete new architecture and market target. The long term strategy for this product was to build a multi-layer, multi-technology and multi-domain product. Currently, exists in the field different platforms for different technology network elements. The new product target was to concentrate on a single platform the management of multiple technology network elements.

The product architecture was divided in three different components distributed over several development units. It was being developed as an off-the-shelf product, i.e., was a general product for the market, although considered, from beginning, customization facilities for potential costumer adaptations.

The product development started in September 2009 and was still in its first version. Therefore, it was a young maturity product trying to reach new potential customers.

3.1.3 - Market

The target market segment for this product is made of customers with complex Telecommunications systems requiring efficient and broad network management.

There was a significant investment in a new product development starting project with no customer identified. It was a market-driven product trying to enter in the market and conquer new customers. During the project two potential customers were foreseen, one with an expected deliver at end of April 2011 and the other at end of August 2011.

In McFarlan's (McFarlan, 1984) portfolio matrix this product is positioned in the "High Potential" quadrant. It was a product with a potential high value in future customer's competitiveness and network management costs efficiency, due to its capability of managing several network technologies on a single platform. It was not essential in current customer's network management as there exists in the field different platforms for each different network technology.

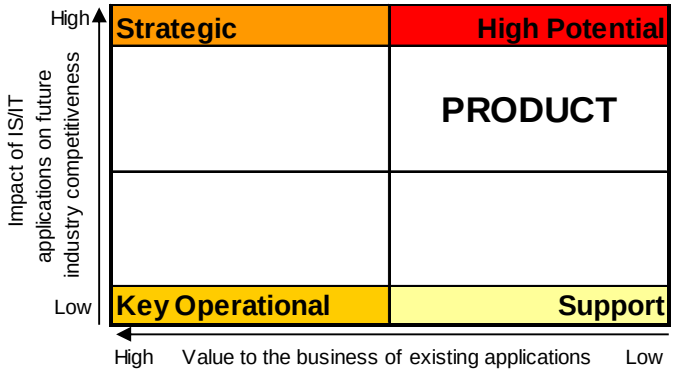


Figure 10 – Product's positioning in McFarlan's portfolio matrix

In terms of constrains, the company was developing a product in a market segment where already exists concurrent product offers with similar targets and features. Also, significant constrains related with the company itself had influence in the project. Company was crossing a strong budget restriction period and operational costs control, focused to deliver good financial results.

3.1.4 - Project Organization

In total, at the end of the project, there were involved in the product development 181 people: 23 Scrum Masters and 158 Team members, distributed over four countries and six different sites.

Focusing on the unit of analysis, Component 3, there were involved, at the end of the project, 113 persons: 12 Scrum Masters and 101 Team members distributed over 14 Teams in four different sites at three countries. The main development site was Portugal Site 1. Additionally there was another site in Portugal, plus one in Poland and another one in India. Portugal Site 1 had five Teams and Site 2 only one Team. Poland site had seven Teams and India site one Team. All Teams had typically eight Team members each and one Scrum Master could be associated with more than one Team. All members of a Team were co-located in the same site, i.e., there were no Teams with members distributed over different sites. The Poland site was an external consultant company contracted temporarily for this specific project.

Time zone difference between Portugal and Poland was +1 hour and with India +5:30 hours (or +4:30 hours in summer, because India does not have Daylight Saving Time[5]). Poland site had -1 hour difference towards Portugal and +4:30 hours (or +3:30 in summer)

[5] Daylight Saving Time (DST): practice of advancing clocks' time by one hour during the summertime. Not all countries apply DST, which is the case of India.

towards India. Therefore, inside Component 3, the maximum time zone difference between sites was 5:30 hours (or 4:30 in summer) difference between Portugal and India. Considering all product components the maximum time zone difference was 7 hours between Portugal and China. The two sites in Portugal were separated by around 250Km distance from each other.

Component 2 had its development unit concentrated in India with 19 persons: 3 Scrum Masters and 16 Team members distributed over 3 Teams. Component 1 had its development unit concentrated in China, but divided in two sites. One of these sites was an external consultant company. There were 49 persons: 8 Scrum Masters and 41 Team members distributed over 9 Teams.

Table 11 provides a complete view of Team and site distribution in the project.

Site/Team Distribution					
	Sites	**Teams**	**People**	**Time Zone**	**TimeDifference at 10AM**
Component 3	**Portugal** **2 sites**	**5 Teams**	**3 SM + 35 TeamM**	GMT+0	Portugal (10am): - in Poland it's 11am - in India it's 15h30 - in China it's 17h00
		1 Team	**1 SM + 9 TeamM**		
	Poland **Outsourcing**	**7 Teams**	**7 SM + 50 TeamM**	GMT+1	Poland (10am): - in Portugal it's 9am - in India it's 14h30 - in China it's 16h00
	India	**1 Team**	**1 SM + 7 TeamM**	GMT+5h30 (no DST)	India (10am). - in Portugal it's 4:30am - in Poland it's 5h30am - in China it's 11:30am
	Totals	**14 Teams**	**12 SM + 101 TeamM = 113 Persons**		
Component 2	**India**	**3 Teams**	**3 SM + 16 TeamM**	GMT+5h30 (no DST)	India (10am). - in Portugal it's 4:30am - in Poland it's 5h30am - in China it's 11:30am
	Totals	**3 Teams**	**3 SM + 16 TeamM = 19 Persons**		
Component 1	**China** **2 sites**	**4 Teams**	**3 SM + 18 TeamM**	GMT+7	China (10am) - Portugal it's 3am - Poland it's 4am - India it's 8:30am
		5 Teams (outsourcing)	**5 SM + 23 TeamM**		
	Totals	**9 Teams**	**8 SM + 41 TeamM = 49 Persons**		
	TOTALs	**26 Teams**	**23 SM + 158 TeamM = 181 Persons**		

Table 11 - Site and Team distribution in the project

Table 12 presents the project's business product area. The overall product management was shared by two persons in Germany, i.e., there was not a single product responsible, this responsibility was shared. Under this product management team, product had three Product Owners (POs) managing each one its own Product Backlog associated with each of the three product components. This means that the overall product had three single independent Product Backlogs managed by three Product Owners. A situation that was against the Scrum

theory which says that only one Product Owner and one Product Backlog should exist (section 2.2.4 -). These three Product Owners were located in Finland, Portugal and Germany. Under these Product Owners, there was a group of Area Product Owners (APO) managing sub-parts of each Product Backlog related with a specific requirements area. In Component 3 case, there were two Area Product Owners located in Germany. To support these Area Product Owners there was another group of Proxy Area Product Owners (PAPO) performing the bridge to the development Teams. Component 3 had two Proxy Area Product Owners, one for each Area Product Owners and both were located in Portugal Site 1.

Product Management 2 persons - Germany	**Component 3**	**PO** Germany	**APO** **APO** Germany	**PAPO** **PAPO** Portugal	Teams
	Component 2	**PO** Portugal	**APO** India	-	Teams
	Component 1	**PO** Finland	**APO** China **APO** India **APO** Finland	-	Teams

Table 12 - Product Owners organization in the project

3.1.5 - Project Scaling

Table 13 presents the Component 3 project scaling in number of Teams and sites along the project duration.

The Sprint number used in the project was a sequential week count number starting on 1st January 2008. Therefore, Sprint #101 means it had its end on week number 101 since 1st January 2008. The difference between consecutive Sprint numbers gives the Sprint duration in weeks.

Team Evolution			
#Teams	**Sites**	**Sprint#**	**Sprint End Date**
2	1 PT1 + 1 PT2	101	03-Dez-2009
3	2 PT1 + 1 PT2	107	14-Jan-2010
6	5 PT1 + 1 PT2	135	30-Jul-2010
9	5 PT1 + 1 PT2 + 3 Pol	137	13-Ago-2010
11	5 PT1 + 1 PT2 + 5 Pol	148	29-Out-2010
13	5 PT1 + 1 PT2 + 7 Pol	151	19-Nov-2010
14	5 PT1 + 1 PT2 + 7 Pol + 1 Ind	157	03-Jan-2011

Table 13 - Team number evolution along the project (Component 3)

Figure 11 presents a chart with the Team number evolution over the project duration for a visual understanding the of the project scaling.

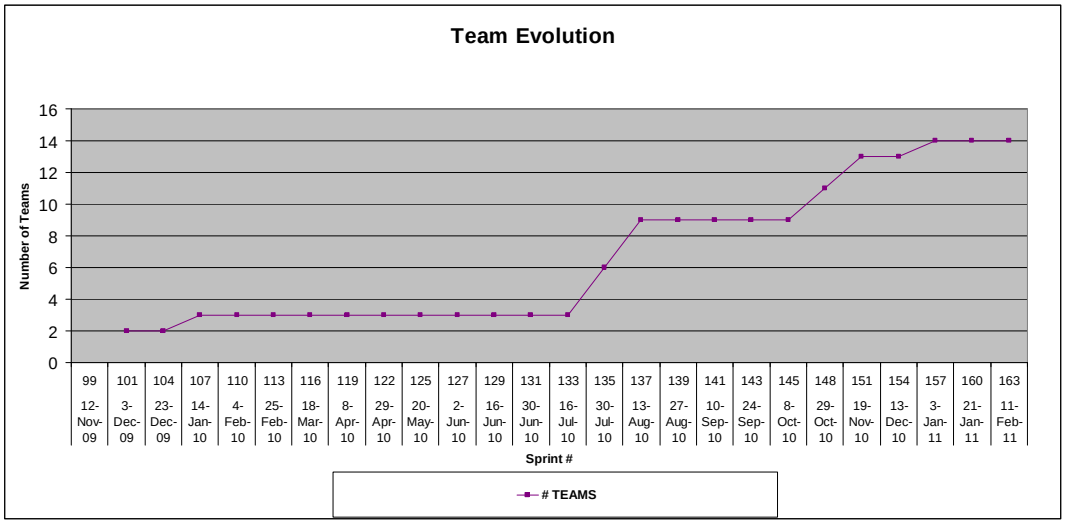


Figure 11 - Team number evolution along the project (Component 3)

The overall product development project started on September 2009. As it can be seen in Figure 11 chart, Component 3 finished its first Sprint (#101) at the end of November 2009 with two Teams involved, one at Portugal Site 1 and the other at Portugal Site 2. A new Team in Portugal Site 1 started on January 2010.

In May 2010 business team and management decided to increase the number of product development people in order to speed up the product release for a potential customer that in meantime was being foreseen. In Component 3 it was decided to increase the number of developers in Portugal Site 1 and to subcontract an external consultant company in Poland. Between May and June it was setup a period of training and knowledge transfer where the experienced people in Portugal site 1 coached the new people, including some people from Poland site. In order to allow a smooth integration of the new people and knowledge spread over all Teams, the experienced two initial Teams in Portugal site 1 were split over five new Teams together with the new coached people. The same integration strategy was applied in Poland, where people trained in Portugal were split over the three new Teams and new people were integrated.

As detailed in Table 13 new people in Portugal Site 1 started being productive on July 2010 (Sprint #135). Poland site started in Sprint #137 on August 2010 with three Teams and kept increasing, being five Teams in October 2010 (Sprint #148) and seven Teams in November 2010 (Sprint #151).

In order to improve the communication with Component 2 in India, it was decided to create there a new Team for Component 3 there. Therefore, on January 2011 a new Team in India was formed to work in Component 3 and keep a closer communication with Component 2 Teams.

In the beginning of March 2011 it was decided, at business level, to perform a product roadmap re-planning. The product scope was significantly reduced and released resources were shift to other business areas with expected faster return on investment. With this scope reduction and resource relocation, it was decided to co-locate the overall product (the three components) in Portugal Site 1. This huge change in organization and product remapping led to a new period of knowledge and development environment transfer towards Portugal Site 1 that lasted for 2 months until the end of April. At beginning of May, due to business priorities re-planning and strong cost reduction pressure, another business and management decision was taken in order to completely freeze the product development. These business and management strategic decisions along the project were non-disclosed decisions and out of this dissertation scope.

An important remark is that the case study covered by this dissertation refers only to the period until when the project was distributed over multiple sites, i.e. until February 2011 (Sprint #163).

Checking Team number evolution in Figure 11 chart it is verified that in around one year was an increase from two to fourteen Teams involved in the project, and from two sites in one country to four sites in three countries. It can also be verified that around half of the project duration were involved only three Teams. On the other half of the project was verified an increase to fourteen Teams, i.e., an increase of eleven Teams.

3.2 - Scrum Practices

Section 2.2.4 - presented a theoretical Scrum method overview, describing its Roles, Events and Values. Figure 5 supported this description with a complete Scrum method diagram.

Now, in this section will be presented how Scrum practices were applied in the project as well as challenges faced due to geo-distribution over different sites. All information in this section refers only to Component 3 project. Further detailed information on the project's Scrum practices can be found in Annex C. A summary table of the project's Scrum practices next described is available in Annex D.

3.2.1 - Sprints

Project started with three weeks Sprints, without synchronization mechanisms between components. Latter, management decided that Sprints would last for two weeks with synchronization mechanisms between the three components. These two weeks Sprint turned out to be not very productive for the project. Sprints setup and meetings overhead left only few days of productive development in each Sprint. Thus, management decided to return back to the three weeks Sprints, but with synchronization mechanisms between components.

3.2.2 - Daily Scrum

Daily Scrum meetings occurred day by day in every Team involved in the project. Each Team decided when it occurred during the working day, but typically the meeting occurred at the end of the morning and lasted a maximum of fifteen minutes. In this meeting each Team member answered the three Scrum questions: “What did you do since the last Scrum meeting?”; “What will you do before the next meeting?” and “Do you have any obstacles?” All members of each Team shared the same workspace and thus there were no issues raised related with distribution in the Daily Scrum meetings.

3.2.3 - Sprint Planning

Sprint Planning was taken in two parts: Sprint Planning part 1 and part 2. Each part was planned with one hour maximum duration and took place in the Sprint’s first day (Portugal: 9:30-10:30/10:30-11:30; Poland: 10:30-11:30/11:30-12:30; India 15:00-16:00/16:00-17:00). Sprint Planning part 2 occurred immediately after part 1.

Sprint Planning part 1 was a joint meeting between Product Owner representatives (Proxy Area Product Owners - PAPO) and two representative members of each Team. The meeting occurred in a common meeting room in Portugal Site 1 connected remotely to the other three sites involved in the project. It was used voice and video conference as well as application sharing software to share Product Backlog (PBL) with every present Team member. Sprint Planning part 1 followed a script of four steps and its details are available in Annex C.

Sprint Planning part 2 was taken at Team level with all Team members present. PAPO presence was not mandatory, but should be available (in person or by phone) to clarify any questions to the Team. Part 2 meeting details can be found In Annex C.

3.2.4 - Joint Product Backlog Refinement (Joint PBR)

Joint Product Backlog meeting occurred on the 2nd Monday of the Sprint from 9:30 until 16:00 in Portugal (10:30 to 17:00 in Poland), i.e., a five hours maximum duration meeting. As the Team from India was very recent in the project and in a learning phase, they were still not participating in the meeting.

Participants were the PAPO team, one representative of each Team, Subject Matter Experts (SME) (people involved in the project with special knowledge on a certain subject) and one facilitator in each site.

Several actions were taken before the meeting as preparation and the meeting itself followed a six step script. Details in meeting preparation actions and meeting steps can be found in Annex C.

Joint PBR meeting occurred normally, but participants felt difficulties to perform technical discussions remotely and not in its mother language. There were also difficulties when the new Teams from Poland start working in the project, because their technical level was still low and the Story Points estimation figures got biased.

3.2.5 - Team-level Product Backlog Refinement (Team-Level PBR)

This meeting occurred at Team level on the 2nd Tuesday of each Sprint, one day after of the Joint PBR meeting and with a maximum duration of three hours.

On Annex C, the details of meeting's five step script are described.

These meetings worked very well with every Team member together around the table with the PAPO nearby. As people were collocated no issues with distribution were reported.

3.2.6 - Sprint Review

Sprint Review was a joint meeting between PAPOs and one representative member of each Team. The meeting occurred in the last Sprint day with a duration of 2:30 hours (Portugal: 11:00-13:30; Poland: 12:00-14:30; India 16:30- 19:00). The meeting took place in a common meeting room in Portugal Site 1 connected remotely to the other three sites. It was used voice and video conference as well as application sharing software to share PBL with every present Team member. Annex C presents the meeting's script in detail.

This was reported as a difficult meeting to perform remotely between all participants. One factor was the time zone difference and different work hours culture. Poland Teams want to leave the office early in the afternoon, around 15h their time, 14h in Portugal. This conflicted

with the lunch time of both sites and a clear agreement in the meeting schedule was never reached between the sites. With the increase number of Teams in the project this meeting started to be impacted and its maximum duration was most of the times exceed. To overcome this schedule and duration issues, the meeting script was continuously being optimized Sprint by Sprint, reaching its final version as described in Annex C.

3.2.7 - Team-Level Retrospectives

These meetings were held right after the Sprint Review meeting, at the end of each Sprint. Their maximum duration was two hours and occurred in every Sprint.

These were team-level meetings where all Team members participate together with their Scrum Master.

The aim of these meetings was to figure out what the Team could do to improve the process in order to increase their productivity. Some problems could be solved only internally to the Team, other problems, that would involve third Teams or the organization, was then taken to the Joint Retrospective meeting to be discussed.

In Portugal these meetings were not very efficient because most of times there was not enough time left for it. This because Sprint Review started to get longer than expected and people wanted to leave office earlier on Friday around 17h. Therefore, must of the times Teams rush on this meeting, degrading its efficiency.

3.2.8 - Joint Retrospectives

These meetings occurred after the Sprint Planning meeting at the first Tuesday of the Sprint with a maximum duration of 1:30 hours. Ideally the meeting should have occurred right after the Team Retrospective and before the Sprint Planning, but due to time zone differences between sites, the closest possible to the Team Retrospective was only after the Sprint Planning. This was not the ideal situation, because they could take decisions in the Joint Retrospective that will have impact in the already planned Sprint.

The joint Retrospective meeting did not have to occur every Sprint. Its realization was decided by Scrum Masters case by case. If the last Join Retrospective discussed issues were already solved or if too many new issues appeared during the Sprint, Scrum Masters could decide to perform the meeting, or not. The idea was not to perform a meeting to discuss the same issues or just a few new issues.

The Joint Retrospectives occurred remotely between Scrum Masters and one representative per Team. One Scrum Master was the meeting facilitator.

The meeting was a forum to bring issues and ideas to continuously improve the process efficiency. During the meeting, facilitator asked the Team representatives “What can we do to enhance the process?” in order to get the process issues experienced by them. Then, all meeting participants discussed issue by issue to try to figure out a solution for them. This generated very interactive meetings difficult to perform remotely.

These joint meetings were held long time in the project, but they start to notice that issues raised there were the same or very close to the ones raised and addressed in Communities of Practices (described next in 7.8.10). Therefore, it was decided to stop the Joint Retrospective meetings and Teams start to report directly issues to the correspondent Community of Practice.

3.2.9 - Scrum-of Scrums

When, in the project beginning, the number of Teams and personnel involved were not so high, it was tried Scrum-of-Scrum (SoS) meetings between all Teams in the product development, i.e., from the three product components. Then, with the increase in the number of Teams and personnel in the project, there were held first a SoS meeting intra-component and then one SoS meeting inter-components where only two Team representatives attended.

SoS meetings were planned to occur every Wednesday intra-component with a maximum duration of thirty minutes and every Friday for the inter-components during a maximum of one hour.

The decision to perform these meetings came from management and not from a Team’s need. Teams felt these meetings were just “more meetings” without much real and useful interest for them. Teams couldn’t reach the meeting aim, which was to share information between them in order that together they could get synchronized and reach a common goal. In these meetings, Teams were just reporting their work status between them and not trying to discuss issues and impediments in order to overcome them. There was verified a very weak remote inter-Team cohesion which compromised the SoS meetings objective. Due to all these facts, these meeting had a very weak participation during the project.

3.2.10 - Communities of Practices (CoP)

These were virtual teams oriented to a specific and specialized theme. CoP members were from several Scrum Teams in the project, specialized or with high knowledge in the CoP

theme. These CoP teams evolved during the project and mainly there were three CoP: GUI; Architecture, and Continuous Integration

The CoP meetings were decided independently by each CoP members without any specified rule. They were totally self-organized virtual teams. Additionally to the meetings they used Wiki to share knowledge and information with the other project participants'.

CoP are not part of the Scrum framework, but they are very common and extremely useful when the number of Teams in a project starts to be significantly high. In this concrete project they turned out to be very efficient and brought much valued added in the discussion and impediment's resolution in the project.

3.3 - Tools

In this section the set of CASE and Communication tools used in the project are presented. The tools were already used in the company for other software development projects. By development environment policies it was decided to keep these set of tools in this project.

3.3.1 - CASE Tools

Several CASE tools were used in the project to support development activities and to address the distributed nature of the project. Table 14 presents the used CASE tools and its purpose on the project.

Name		Purpose
Wiki		- Support knowledge sharing and to document information
Jira with GreenHopper plugin		- Used by Product Owners to manage Product Backlog and to make it available to all Team distributed over the different sites.
Fault Management		- Managing faults on the product
Continuous Integration Solution	**Bamboo**	- For automatic build production
	Subversion	- Open-source version control system
	MPP	- Company internal developed tool used for automatically compile new builds, install them in laboratory machines and run a set of basic tests.
	Rational Robot	- To develop and run automatic tests on builds

Table 14 - Project's CASE tools

3.3.2 - Communication Tools

Several communication tools were available in the project to support the remote communication between Teams. Table 15 presents a summary table with the available communication tools in the project and its purpose.

Name		Purpose
email		- Mostly for asynchronous written communication - Base of remote communication between all teams
Voice Communication tools	**Traditional Telephone**	- To communicate initially with Poland external site
	Internal VoIP tool	- To communication with internal company people and access joint meeting conferences
	Skype	- Started to be used to communicate with Poland external site
Chat Communication	**Internal VoIP tool**	- To communication with internal company people
	Skype	- Started to be used to communicate with Poland external site
Video Communication	**Skype Video**	- To overcome video limitation of company Internal VoIP tool - To overcome lack of face-to-face meeting and improve people remote communication and cohesion
	Halo Room	- To emulate remote people presence - Only available in Portugal Site 1 and India
Application Sharing	**WebEx**	- To share presenters desktop and application during joint remote meetings

Table 15 - Project's Communication Tools

3.4 - Development Performance Analysis

During data collection phase, a set of software metrics from Component 3 project were collected in order to measure and analyse the development performance along the project.

The main used metric was the Burn Up, not only to measure the product size evolution, but especially to measure the development velocity along the project. As it will be presented in section 3.4.1 -, the product scope was always evolving along the project, i.e. the product scope was not constant and therefore, the Burn Down metric did not provide much information and was not used for this study. To measure the development accomplishment ratio, the Descoped metric was used. This metric provides information about the amount of work that was estimated and planned to be performed in a specific sprint, but was not effectively achieved at the end of that sprint. As a quality metric, the number of new Opened, Closed and still Open Faults per sprint was used. In order to provide a comparison bridge with the traditional development typical metrics, the number of Lines of Code of Component 3 was also used as a measure of product size. Table 16 presents a summary about the used metrics used in the case study:

Metric	What is measured	How is measured
BurnUp	- Measures product size - Measures development Velocity	Story Points Story Points / Sprint
Descoped	- Measures amount of work that was estimated and planned to be performed in a specific sprint, but was not effectively achieved at the end of that sprint	Story Points / Sprint
Lines of Code (LOC)	- Measures size of the developed product	Number of effective code lines (without comments, white lines, header...)
Faults	- Measures quality of developed product	Faults Opened; Closed; StillOpen / Sprint

Table 16 - Summary of metrics used in the case study

3.4.1 - Burn Up and Development Velocity

Figure 12 presents the Product Backlog (PBL) evolution in Story Points ("Total Release") as well as the work already completed ("Done Release") along the project. When "Done Release" meets "Total Release" chart, this means that development has completed the planed product release.

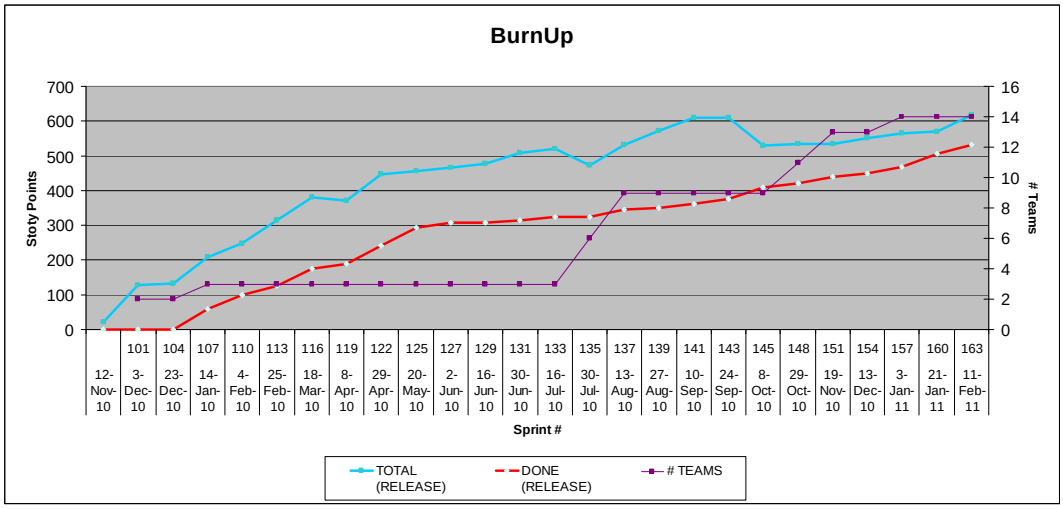


Figure 12 - Project BurnUp chart

It can be verified from Figure 12 chart that the product started to be designed from scratch, i.e., in project's day one Product Owners (POs) did not have the complete view of what would be the product, in this case, Component 3. The PBL was constantly being updated by adding or removing User Stories (USs) and by Story Points (SPs) re-estimation in each Sprint. This explains why "Total Release" chart was not constantly growing, but also means that at each Sprint it provided the best estimated approach of the product number of SPs.

The derivative (slop) of "Done" chart provides the development velocity, i.e., the number of SPs that Teams were being able to produce in each Sprint or week. The chart on Figure 13 shows the development velocity progress in SPs/week along the project duration. The velocity chart was calculated as the average velocity of the last four Sprints. The number of Teams participating in the project was also represented in the chart.

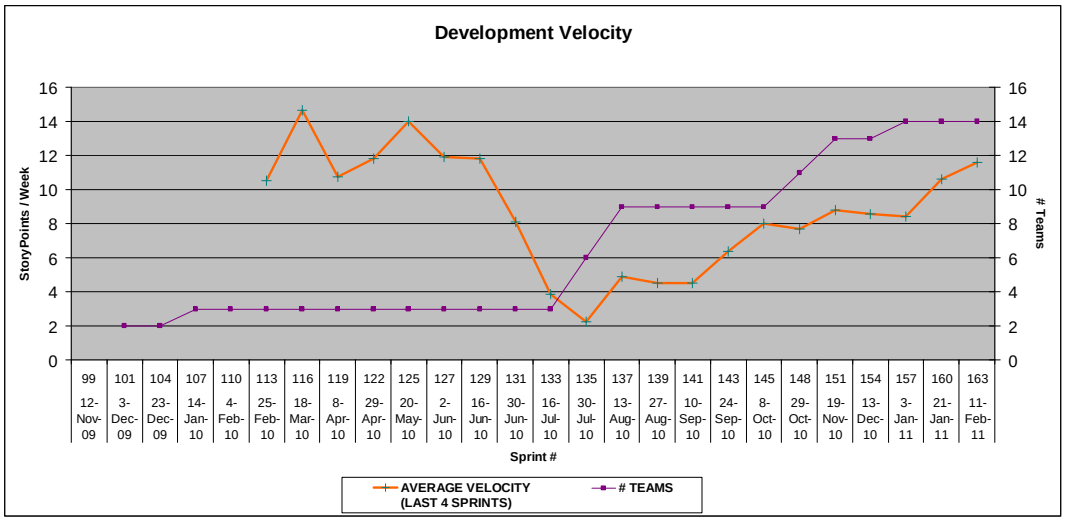


Figure 13 - Project Development velocity chart

Analysing the overall project duration in Figure 13, it can be seen that the development velocity evolution may be divided in three different phases. An initial phase with the best development velocity verified in the project. Next a phase with an abrupt average velocity decrease and finally, a third phase with the average velocity increase at a slow pace.

During initial project's phase, Sprints #107-#125, it was verified a development velocity between 11 and 15 SPs/week with the participation of only three Teams in the project.

Between Sprints #125-#135, was verified an abrupt decrease of around 85% in development velocity. From Sprint #125 on, the number of personnel in the project starts being scaled in Portugal Site 1 and subcontracted Poland site start being involved. In an initial period, was performed new people ramp-up and coaching by the existing Teams in Portugal Site 1. After this learning period, new Teams were formed, starting to be productive and represented in the #Teams chart in Figure 13. Thus, new people ramp-up started on Sprint #125 and first new Teams started to be productive on Sprint #135.

From Sprint #135 on, the development velocity began increasing, although in a low pace and on values significantly lowers than in the project's beginning. It was verified that only at project's end, the average velocity indicator start approaching again the initial average velocity values. As seen in Figure 13 chart, during this period, the number of Teams in the project continued to increase, meaning that new ramp-up and coaching activities were still being performed with impacts in the overall productivity. The last Team to be added was the Indian Team (only one Team), on Sprint #157, meaning also the addition of the fourth remote site involved in the Component 3 project.

Comparing project's initial and final performance was verified that with only three Teams, development was significantly more efficient than with fourteen Teams. This can be clearly verified by comparing the initial and final development's average velocity values and the number of Teams involved (all Teams had typically eight persons involved):

- Initial 3 Teams were producing an average of 12 SPs/week:
  - i.e., 4 SP/week/Team.
- At the end, 14 Teams were producing an average of 9 SPs/week:
  - i.e., around 0,64 SP/week/Team.
- Therefore, for an increase of 366% on the number of Teams involved there was a decreased of 25% on the overall development velocity and 84% on the average Team development velocity.

Table 17 summarizes the above presented development performance facts:

	Initial Performance	Final Performance	Variation
Teams	3	14	**366%**
Avg. Dev Velocity (SPs/week)	12	9	**-25%**
Avg. Team Veloc. (SPs/week/team)	4	0.64	**-84%**

Table 17 - Development Performance Facts

Brooks' Law fits perfectly in the above described phenomenon. It states that "adding manpower to a late software project makes it later" (Brooks, 1995). The two main facts pointed by the author that explain the phenomenon are:

1. The new added people will need time, the ramp-up time, to start to be productive. Also, they will need to be coached by the already productive people in the project in order get trained and catch the work that has preceded them. Thus, while new people are still not producing much valuable work and can even have negative impact (like bugs introduced), the experienced people reduce their productivity in the project.
2. The communication overhead increase with the number of people increase. Communication channels grow with the square of the number of people. Additionally, several people working on the same task will need to get frequently synchronized which will increase the time need to spend in synchronization meetings.

3.4.2 - Accomplishment Ratio

The chart in Figure 14 shows the evolution along the project of Committed SPs by all Teams on each Sprint together with the effective Done SPs on that Sprint. The number of SPs representing the difference between the Committed and Done is known as Descoped and means the number of SPs that were committed and expected to be done, but were not effectively released in that Sprint.

The Commit number of SPs was a Teams' estimation on how many SPs they expect to deliver in each Sprint. As any estimation it had an error associated, but this error was expected to decrease along the project with the Team's growing experience in the product and on the project environment itself.

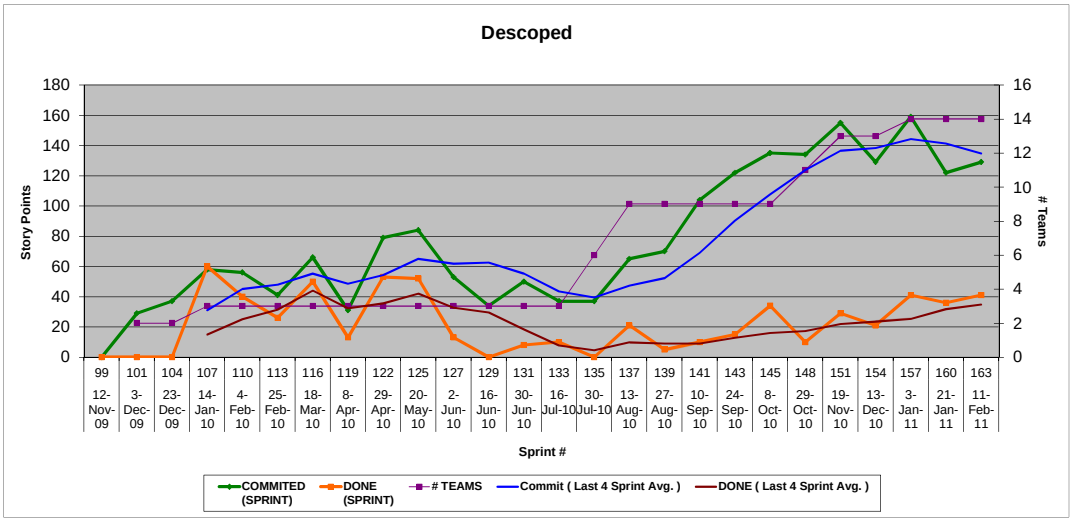


Figure 14 - Project Development Descoped chart

From Figure 14 chart it can be clearly seen that the number of Committed SPs increases with the number of Teams involved. This was expected, because the available development capacity increases with the number of Teams.

It can also be clearly seen in the Figure 14 chart that Done SPs in the Sprint did not follow the same evolution as verified in the Committed SP with the number of Teams increase. This means that the Descoped value rose when the number of Teams also increased, which was not a desirable scenario in a project. This was verified in the chart from Sprint #135 on, i.e., from the start of new Teams in the project.

It was also verified that "Done" progress in project's beginning was positive when the number of Teams was constant. This was expectable and desirable and reflected the Team's

knowledge and performance progress. But, it felt to zero after Sprint #125. From Sprint #133 on, the Done value started to increase slowly and really below the Commit value increase, as seen above.

Comparing the Commit and Done last four Sprints trend curves in Figure 14 chart, it can be seen that both curves were almost parallel and quite close in the project's beginning. After Sprint #125 the curves distance gap started to increase and from Sprint #135 on this gap had a really significant increase. Therefore, the Done positive progress did not follow the same pace as the Commit progress, meaning that the new additional development capacity added to the project was not being effective for product development progress as desired. Both curves only started getting close from Sprint #157 on where the Commit curve started to stabilize and Done curve continued its slow increase. This is another evidence of the Brooks' Law (Brooks, 1995) verification in the project, as described in the previous section.

The next Figure 15 chart represents the Descoped percentage along the project duration. It was also plotted in the chart the number of Teams' progress and the Descoped % trend taking in consideration last four Sprints.

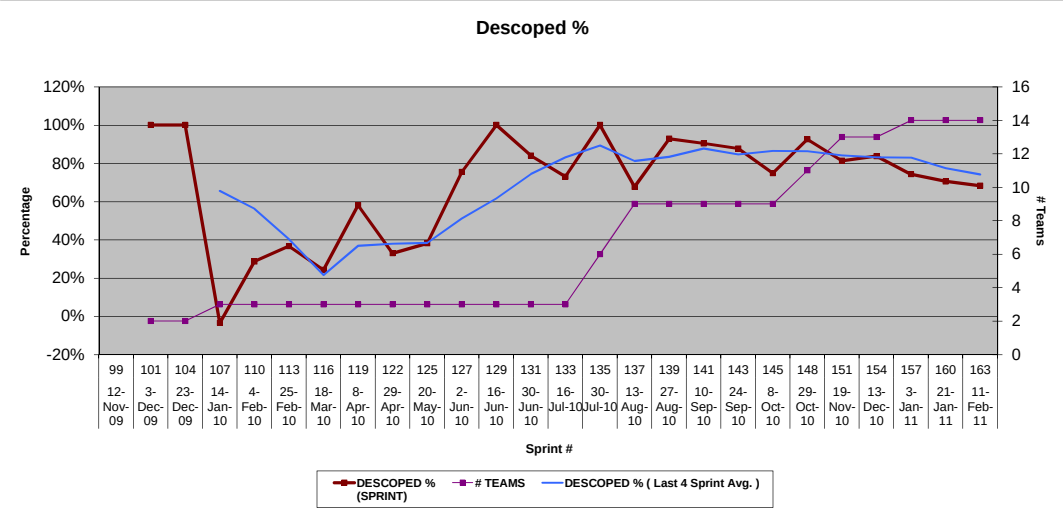


Figure 15 - Project Descoped Percentage chart

It was verified a Descoped value around 40% and with a decrease trend in the project beginning. From Sprint #125 on, the Descoped percentage in the project presented a significant increase until Sprint #135, stabilizing over the 80%. In the final Sprints it was verified again a Descoped decrease.

Descoped % measures the amount of work committed but not effectively done in a Sprint. Descoped depends on how accurate Team's estimation was and therefore, it was not a 100%

accurate measure. Although, it provided an excellent indication of the level of lost or wasted work during a Sprint. Descoped value was influenced by internal and external Team problems'. Internals like possible estimation errors (over or under estimation) and any impediments inside each Team. Externals, like impediments related with enterprise or organization inefficiency that could not be overcome by Teams; or impediments with dependencies with other components; or, for instance, information not available on time to develop the planned Sprint User Stories.

From the chart above it can be clearly seen that the Descoped percentage really increased with the number of Teams on the project. It can also be verified that approaching the end of the project the Descoped % started to decrease reflecting the fact that the number of Teams increase were stabilizing and existing Teams experience and knowledge was improving and being effective.

3.4.3 - Lines of Code

In Figure 16 chart, the number of Line of Code (LoC) evolution in component 3 is presented.

Figure 16 - Lines of Code evolution in Component 3

This graphic is provided just with an informative purpose. Its output reflects the effect of several factors during the project (number of people variation, score re-planning and productivity and quality issues). The number of LoC was not always growing. This is explained due to several code refactoring activities (a constant practice in Agile) for cleaning and optimizing the component code performed during the project. The average number of LoC along the project was around 18 000 LoC, with a maximum of 25 000 LoC near project end. The abrupt decrease in the last part of the chart represented the product scope reduction decided by business management near the project's end (section 3.1.5 -).

3.4.4 - Faults Evolution

The next chart (Figure 17) presents the cumulative number of New, Closed and still Open Fault reports on each Sprint along the project duration.

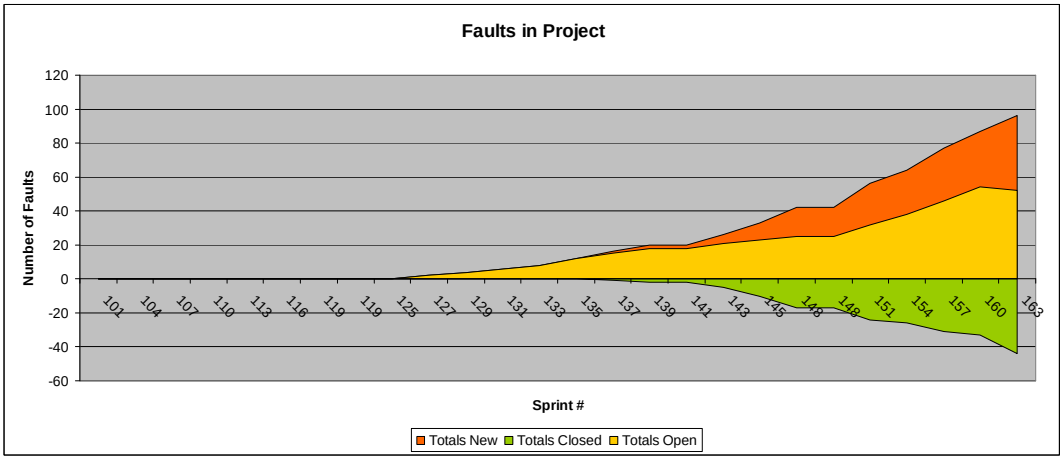


Figure 17 - Cumulative Number of Faults in the Project

It can be seen that there was an initial phase, more centred in architecture and base product development. During this period faults verified in the product were not being registered. From Sprint #125 on, a Quality Assurance team started being involved in the project to handle the product testing activities.

The total number of opened faults along the project duration was around 90 faults, being around half of them (~45 faults) corrected. The fault correction trend in the project's final Sprints presented an increase representing also a decrease trend in the number of still open faults.

3.4.5 - Development Performance Conclusions

From the above analyses it can be concluded that the project's performance was much better at the project's beginning, with less people involved, than at its end, with several people and sites involved.

It was also clearly identified the correlation between the addition of new people and sites in the project and the decrease on the development's productivity and velocity. The descoped percentage was too high during the project scaling period, meaning a low productivity and waste of the available development capacity. These facts suggest the Brooks Law (Brooks, 1995) verification in the project.

On the last project's Sprints, development indicators presented values similar to the project beginning and with a positive recovering trend. Although with a much lower

efficiency (similar indicator values, but with much more people involved), the project development productivity was pointing to a positive trend for the future.

3.5 - Challenges & Impediments

This section will present the interview's results with the reported challenges and impediments in the project. The section will be subdivided according the several topics addressed in interviews. Detailed information about these interviews, as well as sample's characterization, can be found in Annex A. Along the interviews was verified a broad responses consistency provided by the different groups of interviewed participants. In Annex E, a summary table is available with the project's challenges and impediments next detailed.

3.5.1 - Communication

Communication was one of the most and major challenge reported by participants during interviews. The physical separation, together with the lack of face-to-face meetings in the project, was the most referred impediments for a good communication in the project.

Due to a company strict travel costs policy, there were not performed as much face-to-face meetings and team building sessions between remote people as desired and required. Few people in the project had the chance to meet each other. Some met in a company event (between Portugal Site 1 and Site 2), others during the ramp-up and know-how transfer period (between Portugal Site 1, Poland and India). It was clearly pointed by all participants that after meeting face-to-face each other, the communication improved significantly and barriers almost disappeared. Participants referred that the communication between sites was then almost assured between these known people, meaning that other people kept demonstrating resistance to remote communication with people they did not have met before. It was reported that no tools were available to efficiently emulate remote people presence in order to compensate the lack of face-to-face meetings. Skype video started to be used, but although it helped, it was referred as not enough to emulate efficiently remote people presence. Consequences of this lack of face-to-face proximity were, apart from the communication resistance, a mistrust climate, lack of team cohesion and the feeling of "us and them" between the sites.

The language barrier was also referred as an impediment to a more efficient communication, but participants' English level was reported as generally good and up to the needs.

The other main impediment, that had negative impact in communication efficiency, was the Agile culture differences between company internal sites and Poland external company. As Portugal sites and Poland were working in the same code base, the communication needs between them were very high and important. Poland site people were referred as not being very familiar with the Agile culture and ways of working, not understanding why they need to participate in the Scrum meetings as Portugal sites. They also didn't understand the need to develop automatic tests for developed User Stories. These differences had negative impact in the communication with Portugal sites and generated a climate of mistrust and "us and them" between these sites.

Between local Teams was reported a very good climate of trust and open cooperation, without any significant communication impediment. The only impediment reported was related to the significant high number of involved people. This created the "chair inertia" problem where people found resistance to stand-up to talk with a colleague in the other side of the room. Also, it was pointed that sometimes was difficult to know with whom they might talk to solve an issue, because not always was known which problem was already deal by other colleague.

3.5.2 - Office Environment

The overall opinion of all participants was that the office conditions for Scrum practices were good enough, especially at the end of the project.

People reported that in project's beginning the conditions were not good, but they could easily re-arrange the office space and the seat allocation so that every Team were together around the same table and could easily talk between each other. In Portugal Site 1, in the beginning, was not available a dedicated room for the join meetings, but this was also overcome early in the project. In this site, one person reported that would also be desired some small meeting rooms where few people can go in and easily discuss any problem or issue. Portugal Site 1 Teams were also allowed to use flipcharts on the office walls. In Portugal Site 2 it was referred that they had some problems with the meeting room availability and that they could not freely talk in the office (concerning noise) due to the fact that they were in an open-space together with other projects people. In this site, they were not allowed to use flipcharts in the office walls, which was not appreciated by the Team. In India site, apart from the initial office arrangement issues, that were overcome, no other issues were reported.

3.5.3 - Time Zone Differences

This topic was reported as having significant impact in the project's productivity. The main challenge was the reduced common working hours between the sites, inside the same component as well as between components. The most reported consequences of this challenge were the delay in problem solving (which could take 24 hours or even a week), delay in people synchronization, impacts in join meetings and impacts in knowledge transfer. The time zone differences between the different sites were presented in Table 11.

The reduced common working hours challenge was not only related with the real time zone difference between sites. It was extensively reported the different work hours culture between the sites, especially between Portugal and Poland sites. Poland people usually arrived at the office very early in the morning (7h-8h) and left office in the mid of the afternoon (16h-17h). Portugal sites typically arrived later in the morning (9:30h-10:30h) and left in the end of afternoon (18h-19h). Therefore, one hour time zone difference plus two or three hours of different work hours culture, represented an effective time difference between sites of two to three hours, leaving only five to six hours of common working time!

Regarding inter component communication, Portugal Site 1 had to concentrate synchronization and interworking activities with Component 2 colleagues (India) in the morning period. If a problem occurred at the end of the morning, they knew in Portugal that the issue could only be solved in the next day. The interaction and synchronization need between Component 3 and Component 2 was significant and was very difficult due to short common working hours. Communication with China was reported as extremely difficult from Portugal Sites. In Portugal they could only reach Chinese colleagues by arriving early in the morning at the office and for only one or two hours maximum. Communication between them was mostly asynchronous and only supported by email. Every email sent from Portugal need normally 24 hours to get the reply from China and one participant from Portugal Site 1 said: "An email thread with four or five emails takes around one week to be held". Although the asynchronous communication with China was reported as more difficult than with India, the synchronization and interaction need between Component 3 and Component 1 was much lower, which have minimized the problem impact.

Inside Component 3 were reported difficulties in meeting schedule agreements and synchronization between Portugal sites and Poland, due to the different work hours culture. There was also reported by Portugal sites that Polish colleagues commit code just before leaving the office. When these commits broke the build, Portugal could not commit any code

until next day or they had to try to fix by themselves the problem to proceed. India site reported that the short common working hours with Europe sites was an impediment to their ramp-up and knowledge transfer. Also, they reported that short common working hours was a challenge in relation to the continuous integration code commit. India, could commit code that later will break build in Europe sites and they were not in the office to fix the problem.

Not only negative aspects of reduced common working hours were reported. Participants in Portugal referred that when all other sites' colleagues left the office, the communication requests decreased significantly and they could easier concentrate on their tasks, improving productivity. Other positive aspect reported was that they in Portugal could install laboratory machines with a new build and, during their night time, China and India colleagues could already try it out and feedback Portugal sites in the morning.

3.5.4 - Communication Tools

Several impediments were reported by participants regarding the communication tools available on the project.

One reported impediment were the limitations of the company internal VoIP tool. It was reported as enough for one to one communication, but with several limitations for conference meetings as well as not providing video capabilities. As Poland was an external company, this internal VoIP tool could not be used to communicate with them and traditional telephone was initially used instead. To overcome these impediments, Scrum Masters in Portugal sites motivated people to use Skype for their daily remote voice, video and chat communication needs.

The meeting room in Portugal Site 1 had several communication tools' limitations speciality in the project's beginning. Participants reported that the microphone facilities in the room were very weak. Only one microphone was available for joint meetings with several participants. People need to stand up whenever need to speak and it was also difficult when several people need to discuss in the meeting near the microphone. Portugal Site 2 reported that was very difficult to follow a meeting remotely in these conditions, because they stopped listening people's discussion, losing the subject and need to frequently request remote people to repeat. These microphone limitations were mitigated by acquisition of new microphone equipment, although never reached the desired quality. Also, the video facilities in Portugal Site 1 meeting room were not the ideal. Skype video started to be used, but was time consuming to setup it up in every meeting and quality was reported as not good enough.

Apart from this, participants from Portugal Site 2 reported that these video facilities via Skype were an excellent help to their join remote meetings.

Later in the project, Halo Room started to be used. Participants reported that this tool was excellent for remote people presence emulation and a significantly helped in the communication. But, this tool was only possible to be used between Portugal Site 1, India and China. The tool was not available in Portugal Site 2 and Poland. It was also reported that it was not feasible to perform a joint meeting in Halo Room with too many people.

Summing up, communication tools available in the project had several limitations that originated negative impacts in remote communication between sites. Participants desired to have seen more investment in communication tools in the project, especially in sound and video facilities. Also, more investment in remote people presence emulation tools was also reported to have improved the communication in the project.

3.5.5 - Tools Support

Available CASE tools supporting the distributed Scrum practices were generally reported as efficient and enough for the project's needs.

Participants all recognised that Wiki was an essential and extensively used tool for knowledge and information share between all distributed Teams in the project. The only difficulty commonly reported was the information entropy generated in Wiki with the time. Participants reported that the information started to be spread and repeated over several places in Wiki. Outdated and not useful information was kept in place and each Team posted their own information, but other Teams did not know about the update. Searching information in Wiki started to be a difficult task although, although the search engine was reported to be working efficiently. A common feeling was that there should have been defined, in the project's beginning, some rules and guidelines for posting information in Wiki as well as a person should be have been elected to maintain overall information in Wiki updated and organized.

Jira tool was reported as an efficient and central tool for Product Backlog management for several remote Teams. Some Team members reported that although Jira was a good tool, they preferred to have their Sprint Backlog on flipchart at office walls.

The bug tracking tool also did not received a negative appreciation, but participants would prefer to have an easier and lighter tool for bug tracking.

The main problem within the available support tools was the Continuous Integration solution. It was a general opinion that the solution was not working efficiently, especially in

project's beginning, and had significant negative impact in productivity and product quality. The problem with this tool was mainly the test automation scenario. Tests failed to run originating a broken build even if the code was healthy. The problem cause reported was that they started to automate tests late in the project when there were already released too many features. Also, no clear rules for developing tests were defined in the beginning. This originated a huge number of developed tests without sustainability. A taskforce was then created to re-organize and solve the issues with the automatic tests scenario. This measure was a success and the continuous integration system started to work more efficiently, although still with desired improvements.

3.5.6 - Cultural Differences

There were a few impediments raised in relation with cultural differences. The organization had an internal global culture and participants were already used to work in multisite with multi-cultural colleagues.

The common working language was English. Although participants reported that would be more efficient to express in their mother tongue, not significant impediments were reported by the need to communicate in English. Overall people's English level was reported as generally good and enough.

Differences in work hours culture, already described in section 3.5.3 -, was a major reported impediment.

Other impediment reported was the differences in enterprise culture. Poland site was an external consultant company and were reported to have a short term mind-set. They tried to demonstrate fast work done instead of working in long term and for the overall project goal. Also was refereed that they didn't have a good Agile culture. Teams were led by team leaders instead of being self-organized and coached by the Scrum Master.

India was reported as not having cross functional teams, i.e., they had specific persons dedicated to development, test, and installation functions. A hierarchical structure was also seen in India. Teams had more like a team leader than a Scrum Master coach, which is not a very Agile way of working.

3.5.7 - Team Cohesion and Commitment

A common reported feeling between all interviewed participants in Component 3 was the lack of cohesion and commitment felt in the overall project. Between local Teams the relationship was reported as very good and close. The main issue was naturally between

people in different sites. In the project initial phase, when were only few Teams and only two sites involved in Component 3 development, team cohesion and commitment was reported as being much stronger than when too many Teams and locations were involved.

Between Portugal two sites, in the beginning there was some distance and communication resistant between people, but this have disappeared after they met face-to-face in a company event. The cohesion between these two sites was reported has much stronger than with other sites due to cultural proximity.

The main problems were reported to be with the Poland site. Their working mind-set was very different from the one practiced internally in the company. It was reported that they did not have a very Agile way of working. They created dependencies between their and Portugal's code and were resistant to changes. They were contracted for short term and was desired that they would work like internals to the company. Portugal sites never felt they were working as the same team for the same goal. Their commitment to the overall project success was reported as not existing, being their main goal to demonstrate work done in a short term to justify their contract. Some participants in Portugal clearly had the "us and them" feeling.

Between components, the cohesion was reported as not being the healthiest. Although all people were working for the same product, they had the feeling of being working for three different products with dependencies between each other. Each component was concentrated in performing its own work and not in archiving all together the product goal. It was reported that was difficult to know what was happening inside each of the other components and a very weak synchronization inter-component was verified during the project. This lack of cohesion between different components was pointed to be caused by the shared product responsibilities, i.e., three Product Owners managing three different Product Backlogs for three different development units.

3.5.8 - Division by Components

As already described, the product development was divided into three difference components. This organization was commonly reported by participants has not being very efficient and especially not very Agile. User Stories defined by Product Owners were seen as E2E features, i.e., complete customer features that could cross vertically all product components. Also, the division by components created dependencies and interfaces between components and Teams, breaking the Agile open communication and collaboration. When a User Story required changes in more than one component, participants reported that they

need to start conversations with the other component to describe the requirements and then wait until the development was finished to complete their User Story. This created a serial development and consequently longer features' releases. One participant referred that this division by components created like a hidden waterfall inside Scrum. It was also referred the difficulties to coordinate all the three component releases for the product, as well as, load balancing the work load between components. Could be the case that one component was overloaded and other component had free capacity, but there could not be capacity exchange. The only positive aspect referred was that this division decreased the need of communication, because less people were working in the same code base.

When asked for suggestions on how the product development should be divided there was a consensus that division by components was not the best solution and it should be division by E2E features. With this, participants referred that the project organization would be more aligned with the Agile principles. User Stories could be then developed truly E2E. Teams would be able to work on the code base from top to bottom and did not need to request and wait for other components' development. It was also referred that with this approach, only one Product Owner and one Product Backlog should exist for the product. Limitations to this E2E features approach were also reported by participants. They raised concerns regarding the required extensive code knowledge to perform E2E feature development, as well as, the higher communication needs due to a larger common code base between Teams.

3.5.9 - Large Number of Personnel

Component 3 development had its initial phase with only three Teams distributed over two sites in Portugal. Participants reported that during this project phase they had very quick synchronization meetings and team cohesion was good. With the increased number of people involved in the project, communication channels multiplied exponentially and communication overhead was reported as being huge. This report is in line with what was described in section 3.4.1 -Brooks' Law (Brooks, 1995) was extensively verified in the studied case and confirmed with the participants' reported experience.

The people scaling pace in the project was referred as being too high and not done in a sustainable way. Original project's people did not have the time to consolidate their knowledge and couldn't then efficiently coach the new people. Participants referred that the product architecture and development environment, in particular the continuous integration solution, was not stable enough to support a huge and fast project increase. An interviewed manager referred that would have been desired to have more Teams working in Portugal Site

1 than to scale people in Poland external site. At the end of the project, there were more people and Teams in Poland external site than in Portugal Site 1 (the project's main development site).

It was suggested by one participant that project scaling should start first with few co-located Teams to achieve a stable and well defined architecture as well as creating a very stable development environment. Only when these conditions were achieved it should be started a planned sustainable project scaling to other sites and involving more people. Other participant shared the opinion that should be analysed by the company if, instead of subcontracting cheaper and many new developers, hiring less but more experienced developers on the internal sites wouldn't be a more profitable scenario for the company.

3.5.10 - Multisite

The unanimous response from participants to this topic was "No, thanks!". When asked what were the positive aspects of a multisite organization in the project all participants demonstrated extreme difficulties to answer. One positive point referred was the ideas' diversity richness to solve problems, achieved by having different ways of thinking and working in the project. But this could also be achieved by having people from different locations and cultures working at the same place and not necessarily in multisite. The other positive point was the theoretical capability of 24 hours development, but no one felt that this would be possible to reach in practice in the project.

The main negative aspect referred was the communication drawbacks raised by the distance separation. Too much communication channels, huge communication overhead and synchronization need were pointed as some of communication drawbacks. Agile principle of open and close communication was referred to be negatively impacted. The physical distance separation was also pointed as an impediment to the team cohesion and team building, generating a mistrust climate between remote site Teams. Remote joint meetings were reported as very difficult to be held, because of the high interactivity nature of these meetings. The communication impediments raised by physical distribution were reported as causing delays in problem resolution with impacts in final product quality and time-to-market.

All participants demonstrated a significant low motivation about working in a multisite environment. They referred to have a high overhead of non-development tasks (huge communisation, synchronization and meeting's needs), getting a low productive and inefficiency feeling. The reduction on the number of involved sites in the project was commonly seen as a benefit. Communication would highly improve between people as well

as team cohesion, motivation and commit to achieve together a common goal. Communication overhead reduction, efficient synchronization and cohesion between Team members would generate a faster and more efficient project evolution contributing to a better product quality and time-to-market schedules. No travel costs to perform face-to-face meetings as well as less cost in communication tools needs together with no cultural and time-zone differences issues were seen as benefits of project co-location.

3.5.11 - Business Organization

Although out of the scope of this study, is here referred some feedback received during the interviews regarding the business organization in the project.

It was referred by Proxy Area Product Owners that Program and Product managers had a waterfall mind-set instead of being aligned with the Agile way of working. This originated difficulties in communication and alignment between business teams and R&D teams.

As already described in previous section 3.1.4 -, the product did not have a single responsible person. This responsibility was shared among two product managers. The product was divided in three components managed by three different Product Owners with three single Product Backlogs. Development was also divided into three different development units. This spread organization and shared responsibilities were not seen as the most efficient and was reported has not promoting a one product vision. Participants felt that there were working for three different products with dependencies between them and not for a single product focused on the same goal.

3.6 - Case Study Conclusions

According to Hossain et al. (2009a) and Jalali et al. (2010) literature review results, described in section 2.4 -, this case study was an example of an empirical study about a globally distributed project using Scrum, performed in an industry,. The dimension scale of the studied project was significantly larger than most of the studies available in the scientific community. According to the Jalali et al. (2010) classification in Table 9, the studied project was a large (more than fifty people involved) and long (more than seven months) project. Considering Hossain et al. (2009a) Table 8, the studied project, unlike the majority of the studies, had more than three sites involved and more than two Teams involved (four sites in three countries, fourteen Scrum Teams involved). The Team size was significantly lower than the majority (typically eight persons Teams unlike the majority, where Teams had more than

twenty five persons). The collaboration mode was inter-organization (as one of the sites was an external subcontracted company) although, the majority of studies referred an intra-organizational mode. Also unlike the majority, there was verified overlapping time between the sites (although in some of the cases only a few hours of overlap time was verified). The majority of the studies reported to have distributed Teams, as seen in Jalali et al. (2010) Table 9. In the studied project, Scrum Teams were not distributed. They were co-located Teams, i.e. all members of the same Team were together in the same place.

Challenges

The challenges identified in the case study were extensively in line with the ones referred in Hossain et al. (2009b) conceptual framework.

The main challenge faced was clearly the communication. This is the major and typical challenge faced in distributed software development, as identified by Hossain et al. (2009a) and Jalali et al. (2010). Several factors contributed to this communication challenge in the studied project. Unexpectedly, as seen in section 3.5.6 -, the use of English as a common language was not a strong communication limitation, although communicating in mother tongue would have improved the communication efficiency. The lack of face-to-face meetings between people from different sites was the major factor that has led to resistance in communication. This fact was not compensated with efficient communication tools for remote presence's emulation. Although there were available multiple communication tools, they had several technical limitations which decreased the user experience and did not fully cover the distributed communication needs. Time-zone differences problem, augmented by the different work hours culture between sites, caused asynchronous communication and difficulties in achieving meetings schedule agreements.

Lack of group awareness and commitment was also identified as a major challenge. Described in section 3.5.7 -, people did not felt as working for the same product and not committed to achieve a common goal. The product development was divided in three components with three single independent Product Backlogs, which caused that each development Team was working for their component goal and not for the product as a whole. This product organization went against the Scrum theory which says that only one Product Owner and one Product Backlog should exist (section 2.2.4 -).The different enterprise's culture, together with the lack of face-to-face meetings, has led to a mistrust climate between different remote sites. This had impact in the overall commitment and, as already referred before, in the communication efficiency.

The large number of people and sites involved represented also a challenge in the project. As seen in section 3.1.5 -, the project has started with only two Teams in two sites in the same country, but has fast grown to fourteen Teams in four sites in three countries. Before starting the project scaling, the original project's participants did not have the time to consolidate their knowledge as well as the product architecture and development environment. The new people coaching and knowledge transfer, by the original project's people, have had a huge negative impact in the development productivity, as analysed in section 3.4 -. This huge project scaling increased significantly the project's entropy, decreasing even more the communication efficiency and augmenting Teams' synchronization need. All these facts points to the Brooks' Law (Brooks, 1995) verification in the project (section 3.4.1 -).

Tools available were generally enough to support the distributed Scrum practices. The only limitation was the continuous integration solution during the first half of the project duration. The solution presented problems with test automation and represented an extreme decrease in the project productivity. As soon as the problem was overcome the continuous integration solution contributed positively to the distributed development productivity.

A challenge, referred in Hossain et al. (2009b) conceptual framework, which was not verified in the studied project, was the lack of collaborative office environment. The office environment in the several involved sites could be efficiently adapted to the Scrum practices needs.

Strategies

Several strategies identified in the literature review were not verified, while others were.

Paassivaara et al. (2009) and Hossain et al. (2009b) refer frequent visits as a practice used to build and maintain trust as well as to enhance collaboration. Due to a strict travel costs policy, only a very few face-to-face meetings took place along the project duration. This clearly had negative impact on the communication efficiency, trust and team cohesion as the participants reported.

Synchronization of work hours is also referred by Paassivaara et al. (2009) and Hossain et al. (2009b) as a practice in Global Software Development to maximize the overlapping work hours and ensure a constant communication. As referred in section 3.5.3 -, the overlapping work hours between the involved sites was not very extended, either due to countries time-zone difference and different work hours culture. In the project there was no synchronization work hours performed. Each country involved had its own work hours culture, which has decreased the maximum possible overlapping hours. This has caused difficulties in

communication (asynchronous communication), delays and difficulties in reaching meeting schedule agreements.

Splitting Teams by features/functions is a strategy referred by Hossain et al. (2009b) in order to address the large number of personnel involved in a project. As described in section 3.5.8 -, the overall product development was divided by components. This permitted a reduction of communication need between components' Teams, but generated a significant number of inter-component dependencies that caused delays and a serial development. The common opinion by participants was that the project would have extensively benefited if product development was divided by E2E features. This would allow a more Agile approach, as the Scrum User Stories are seen as E2E functionalities, and developers would not be dependent on other Teams to complete their User Stories.

Some strategies referred by Hossain et al. (2009b) and Paassivaara et al. (2009) that were verified in the project. In order to face asynchronous communication and increasing number of Teams challenge, Hossain et al. (2009b) refer that Scrum Teams should be kept local site based Teams. This practice was verified in the project since its beginning. Regarding challenges with communication and support tools, there were available multiple ways of communication as well as tools to support Scrum practices, as referred also by Hossain et al. (2009b) and Paassivaara et al. (2009) (although, as already described before, there were several limitations on the available tools that have decreased its expected efficiency). Regarding the office environment, the available space could have been re-arrange in order to keep each Team together around the same area as well as having dedicated joint meeting rooms.

A strategy successfully applied in the studied project, that was not found in the literature review was the Communities of Practices. As described in section 3.2.10 -, this practice was very efficient and brought much valued added in the discussion and impediment's resolution in the project. As an extension to Scrum practices, Communities of Practices could be an enhancement proposal to the Hossain et al. (2009b) conceptual framework, used to address issues that create impediments to the Teams and where Teams work together to mitigate those issues.

4 - Conclusions

The applicability of Agile methodologies in Global Software Development is far from being a completely solved and consolidated theme.

Literature review has identified a significantly increase on the theme's interest as well as on the number of published studies in the scientific community over the past few years. Despite of this positive fact, was concluded that there is still no solid empirical evidence about the successful applicability of Agile methods in Global Software Development. Several of the available studies reported success, but they lack on details about the studied projects and their contextual environment in order to reach solid conclusions. Also, the available scientific knowledge about this theme was found to be spread, not structured and consolidated. Hossain et al. (2009b), in their systematic literature review, provided a first attempt with a consolidated conceptual framework about challenges faced and strategies to address those challenges when applying Agile methods in Global Software Development.

The challenges faced in the studied project were extensively in line with the ones referred in Hossain et al. (2009b) conceptual framework. The main challenges faced were communication, reduced overlapping working hours, lack of group awareness and overall commitment, large number of people and sites involved. Limitations on the available tools represented also a significant challenge. Continuous integration solution was not working efficiently and communication tools presented several limitations and were not efficient to emulate remote people's presence.

On the other hand, several of the strategies presented in the Hossain et al. (2009b) framework were not verified in the project. Frequent visits, synchronized working hours, efficient communication tools and Teams divided by features were strategies not followed in the studied project.

From the project's development performance analysis, it was concluded that the project was significantly more efficient in its initial phase, with just few Teams seated in two sites in the same country, than in its final stages, where several Teams were involved and spread over four sites in three different countries.

This dissertation contributed to provide a recent view of the current available knowledge about Agile methodologies practice. Furthermore, this dissertation contributed with a case study of a larger and longer project than the majority of the available studies in the scientific

community. The studied project's environment was detailed as much as possible in order to provide a solid reference. The case study results could contribute to validate Hossain et al. (2009b) conceptual framework, comparing the challenges and strategies presented in the framework and the ones identified in the studied project.

4.1 - Answer to the Research Questions

What are the challenges faced when applying Agile methods in a Global Software Project?

The main challenge faced in distributed Agile development is communication. Being Agile based on open, close and constant communication, it is crucial that communication between people in the project is efficient. In a distributed environment, communication is affected by people's physical separation and is augmented when people do not know each other personally. Cultural differences, especially not expressing in mother tongue, represents also a barrier to efficient communication. A large number of people and sites represent a high number of communication channels raising even more the communication challenge. Consequences of these barriers are an inefficient communication, contributing to a mistrust climate and lack of group awareness.

Working with different time-zone represents also a challenge in distributed Agile development. The reduced overlap working hours cause asynchronous communication originating delays in problem solving and information share. Also, difficulties to schedule joint meetings are raised.

Cultural differences place several challenges. The need to express in a common language, normally not the mother tongue for most of the involved people, is a barrier to efficient communication. Differences in work hours culture may decrease even more the overlapping work time. Different work mind-sets and enterprises culture is a challenge with impacts in communication which may originate lack of trust and group awareness within the project's people.

Large number of personal and involved sites may represent a huge challenge. Communication channels increase with the square of the number of people. Also, the overhead for synchronization between people will increase. As communication is already itself a challenge, increasing its need in the project will only augment even more the challenge. Project's entropy increase and people tend to get demotivated with high non-productive overheads.

What are the strategies to address the impact of these challenges in the development's project efficiency?

As communication is the main challenge identified, mitigating its impact is the main strategy. Remote people meeting each other face-to-face, at least once, is essential for breaking communication barriers and raising trust level between them. This will help reaching group awareness and break the feeling of "us and them". Synchronizing working hours is also an essential strategy to maximize the overlapping working hours. Setting up co-located Scrum Teams, i.e. Team members together in the same place, will keep their communication local and very efficient. This practice decreases the remote communication need in the project by keeping intra-Team communication local.

Support and communication tools are essential in a distributed Agile. Their efficiency is essential to mitigate communication barriers, support distributed productivity and knowledge share. Wikis are essential for remote knowledge and information share. An efficient Continuous Integration solution will support the distributed development activities and significantly contribute to its productivity. Product Backlog support tools, like Jira, are also essential to make it available for all people in the project and keep them synchronized with Product Owner. Good quality voice and video communication tools are essential for remote communication and remote's people presence emulation.

Good office environment conditions need to be provided in order to assure efficient Scrum practices. Scrum Teams need to be able to seat together in the same place or room and the possibility to freely use flipcharts on the walls is usually appreciated by them. Dedicated meeting rooms with communication tools ready to be used are also essential.

In large product development with a high number of people involved, Teams subdivision should be treated carefully. Although dividing development Teams by components may decrease the communication need, it will create borders and dependences between them. Development will tend to be serial, meaning that a hidden waterfall may be created inside an Agile distributed project. Diving Teams by End-to-End features will be more in line with Agile, as User Stories are typically seen as End-to-End customer features. With this division developers are able to work vertically in product code and not dependent on other components to finish their User Story in the Sprint. This approach however has a higher communication need as the common code base is also higher. Therefore, the Team subdivision should be carefully analysed before project start.

Project scaling, in number of people and sites, should be carefully planned and performed in a sustainable manner. Project should start with a small number of Teams, co-located or on a very close distribution. It should be assured that product architecture and development environment is well defined and stabilized before starting scaling the project. Distribution over too many sites, especially separated by several time zones, may raise significant challenges as seen before. Therefore, the Agile project distribution benefits should be carefully analysed before implementation.

4.2 - Limitations

This study was limited to one unit of analysis associated with just one of the three product's components. It would have been desired to study the project as a whole, with perspectives from every component as well as from the business areas. This was not possible due to time and resource limitation in performing this dissertation. Also, within the unit of analysis it was not possible to contact Poland participants due to the fact that they were from an external consultant company.

4.3 - Future Work

An overall consolidation work regarding the applicability of Agile/Scrum methodology in distributed development should be continued in order to achieve a reference framework. This framework would represent the distributed Scrum practices, challenges typically faced and the critical success factors to apply Scrum practices in a distributed environment. Companies whiling to apply Agile/Scrum in their distributed software development project will them be able to better analyse and plan their implementation in the most efficient way.

References

ABRAN A. & Moore J.W. 2004. Guide to the Software Engineering Body of Knowledge (SWEBOK®). IEEE Computer Society 2004 Guide, Angela Burgess, 2004

BROOKS Jr. F.P. 1995. The Mythical Man-Month: Essays on Software Engineering, Anniversary Edition. Addison-Wesley, 1995

CMMI Team. 2010. CMMI® for Development, Version 1.3 . CMU/SEI-2010-TR-033, 2010

DUBINSKY Y., Talby D., Hazzan O. & Keren, A. 2005. Agile metrics at the Israeli Air Force. Proc. Agile Conf, 2005, pp.12-19

HARTMANN D. & Dymond R. 2006. Appropriate Agile Measurement: Using Metrics and Diagnostics to Deliver Business Value. Agile Conference 2006, 2006, pp. 6 pp. -134

HERBSLEB J.D. & Moitra D. 2001. Global Software Development. IEEE Mar/Apr 2001 Vol. 18, Issue: 2, 2001, pp. 16 - 20

HOSSAIN E., Babar M., H. young Paik & Verner J. 2009a. Using Scrum in Global Software Development: a Systematic Literature Review. Fourth IEEE International Conference on Global Software Engineering, 2009, pp. 175-184

HOSSAIN E., Babar M., H. young Paik & Verner J. 2009b. Risk Identification and Mitigation Process for Using Scrum in Global Software Development: a Conceptual Framework. Asia-Pacific Software Engineering Conference, 2009, pp. 457-464

JALALI S. & Wohlin C. 2010. Agile Practices in Global Software Engineering – A Systematic Map. 2010 International Conference on Global Software Engineering, 2010

KITCHENHAM, B. & Charters, S. 2007. Guidelines for Performing Systematic Literature Reviews in Software Engineering. School of Computer Science and Mathematics, Keele Univertisity, 2007

LARMAN C. 2003. Agile and Iterative Development: A Manager's Guide. Addison Wesley, 2003

LARMAN C. & Vodde B. 2008. Scaling Lean & Agile Development: Thinking and Organizational Tools for Large-Scale Scrum. Addison Wesley, 2008

LARMAN C. & Vodde B. 2010. Practices for scaling lean & agile development : large, multisite, and offshore product development with large-scale Scrum. Addison Wesley, 2010

McFARLAN F.W. 1984. Information technology changes the way you compete. Harvard Business Review, May-June 1984, pp. 93-103

MILLS E. E. 1988. Software Metrics. SEI Curriculum Module SEI-CM-12-1.1, SEI Curriculum Module SEI-CM-12-1.1, 1988

PAASIVARA M., Durasiewicz S. & Lassenius C. 2008. Distributed Agile Development: Using Scrum in a Large Project. International Conference on Global Software Engineering, 2008, pp. 87-95

PAASIVARA M., Durasiewicz S. & Lassenius C. 2009. Using Scrum in Distributed Agile Development: A multiple Case Study. International Conference on Global Software Engineering, 2009, pp. 195-204

PETERSEN, K. & Wohlin, C. 2009. Context in industrial software engineering research. Proc. 3rd Int. Symp. Empirical Software Engineering and Measurement ESEM 2009, 2009, pp. 401-404

RAMESH B., Cao L., Mohan K. & Xu P. 2006. Can distributed software development by Agile?. Communication of ACM, October 2006, Vol. 49, No. 10, pp. 40-52

RUNESON P.& Host M. 2009. Guidelines for conducting and reporting case study research in software engineering. Empirical Software Engineering 14(2), 2009, pp. 131–164

SOMMERVILLE I. 2007. Software Engineering, 8th Edition. Addison Wesley, 2007

WESTFALL L. 2005. 12 Steps to Useful Software Metrics. Team, T. W. (Ed.), 2005

YIN, R. K. 2003. Case Study Research Design and Methods, Third Edition. SAGE Publications, International Educational and Professional Publisher. Thousand Oaks. London, UK

Annex A – Semi-Structured Interviews

Semi-structured interview questions were prepared based on the Hossain et al. (2009b) conceptual framework with the addition of two open questions at the end. Interviews were held by one researcher performing questions and registering the answers in a document, as well as, recording interviews in audio for post re-check.

The interviews were started by presenting first an overview of the study theme and objectives, as well as a questionnaire overview. Each interview had duration of around one hour and was performed in three out of the four sites involved in Component 3 development.

In Portugal Site 1 there were interviewed eight persons: one Scrum Master, one Manager, two Proxy Area Product Owners and four Team members (each from a different Scrum Team). In Portugal Site 2 there were interviewed two persons: one Scrum Master and one Team member. In India site it was only possible to interview one Scrum Master.

All interviews in Portugal Site 1 where performed face-to-face with interviewees while Portugal Site 2 and India interviews were performed remotely via VoIP. People from Poland site were not possible to be interviewed as they were from an external company and interviews were not allowed.

Next is presented the semi-structured interviews materials created and used:

- Study context and interviews objectives presentation;
- Matrix with interview's questions subject by Scrum role;
- List of questions per Scrum role interviewed.

Also presented a table with interviews' data summary:

- Site interviewed;
- Role interviewed;
- Date of interview;
- Duration of interview;
- Total number of interviews and interviews' duration.

"Applicability of Agile Methodologies in Global Software Development Projects"
- A Scrum Case Study -

Interviews with distributed Scrum project participants

In this document it is presented a set of questions that will guide the semi-structured interviews to be performed to the participants in a distributed Scrum project. This set of interviews will be a source of information for the single Case Study being performed for a masters dissertation which aims to study the applicability of Agile methodologies in Global Software Development Projects.

The dissertaton works to contribute for an answer to the problem of What are the Critical Success Factors to efficiently apply Agile methods in Global Software Development Project?

As objectives the dissertation address the following two questions:

1- What are the challenges faced when applying Agile methods in a Global Software Project?
2- What are the strategies to address the impact of these challenges in the development's project efficiency?

This set of interviews aim to get participants information and their perception of the challenges faced during the previous project organization where it was distributed over four sites.

The questions are specific for the different Scrum participants in the distributed project: Team, Scrum Master, Product Owner and additionally, a Manager.

Interviews are planned to be performed to:

5 - Team members (one per Scrum team)
2 - Scrum Masters
1 - Product Owner
1 - Manager

and it is expect that each interview will take around 45 minutes. (Total interviews ~7 hours)

Thank you for your time and contribution,
João Sampaio

April 2011

ICSTE - MGSI 2009/2011

Supervisior: João Barreiro
ISCTE Supervisors: Professor Mário Romão / Professor José Cruz Filipe

Questions' subjects	Team	Scrum Master	Product Owner	Manager
1. Asynchronous				
Time-Zone differences challenges	x	x	x	x
Practices/Measures to address time-zone differences challenges		x		x
2. Lack of Group Awareness				
Communication challenges	x	x	x	
Face to Face meetings	x		x	x
Team cohesion and commitment	x	x	x	x
Knowledge level of Scrum		x		
3. Poor communication Bandwidth				
Communication impediments with available communication tools	x	x	x	x
4. Lack of Tool Support				
Efficiency of available tools (Wikis, Jira, ...)	x	x	x	x
People correct usage of available tools	x	x		
5. Large Number of Project Personnel				
Challenges with the current Scrum team assignment (by components)	x	x	x	x
How current Scrum team assignment facilitates communication between remote team members	x	x		
How should Scrum teams be assigned regarding the software architecture to increase efficiency	x	x	x	
Benefits and disbenefits of current Scrum team assignment (by components)				x
6. Lack of Collaborative Office Environment				
Impediment with office environment	x	x	x	x
7. Increased Number of Sites				
Positive and negative aspects of project distributed over different site	x	x		
How would project benefic if it was not distributed	x			
Is the current number of sites involved in the project efficient		x	x	x
8. Open questions				
Main difficults which negatively impact development productivity	x	x		x
Main difficults which negatively impact product quality		x	x	x
Main difficults which negatively impact customer satisfaction			x	
Proposal to overcome difficulties mentioned	x	x	x	x
TOTALs	**15**	**17**	**13**	**13**

Team
1. Asynchronous
1.1 Can you list the main difficulties you face in your daily basis work regarding time-zone differences?
2. Lack of Group Awareness
2.1 Which are the communication challenges that you face with remote Scrum teams? How does the communication challenges differ regarding the different remotes sites?
2.2 Do you also face communication challenges with your local Scrum teams? If yes, which are the main challenges?
2.3 How do you feel team cohesion and commitment between your teams and the multiple remote teams?
2.5 Which are the most significant cultural differences that you face when working with remote colleagues?
2.6 How do you think meeting face-to-face remote colleagues would help your daily basis working communication?
3. Poor communication Bandwidth
3.1 Which are the most significant impediments in your remote communication related with the available communication tools?
4. Lack of Tool Support
4.1 How do you think the available tools (Wikis, Jira, Cont. Integration, …) are efficient and enough to support the distributed Scrum practices?
4.2 In your opinion, people correctly use the available tools for the distributed Scrum practices? What do you advise to enhance its use?
5. Large Number of Project Personnel
5.1 Regarding your experience, what are the main difficulties that the current distributed Scrum team assignment by components raise in your daily work?
5.2 How do you think the current distributed Scrum team assignment by components facilitates the communication between remote team members?
5.3 In your opinion, how should Scrum teams be assigned, according to the actual product software architecture, in order to increase your productivity?
6. Lack of Collaborative Office Environment
6.1 Which are the most significant impediments you face with your office environment regarding Scrum practices?
7. Increased Number of Sites
7.1 In your opinion what are the positive and negative aspects of having the project distributed over multiple sites?
7.2 In your opinion, how would the project benefit if it was not distributed over different sites?
8. Open questions:
8.1 Regarding your experience can you describe the main difficulties in the project that impacts negatively your development productivity?
8.2 Which recommendations do you propose to overcome the above mentioned issues?

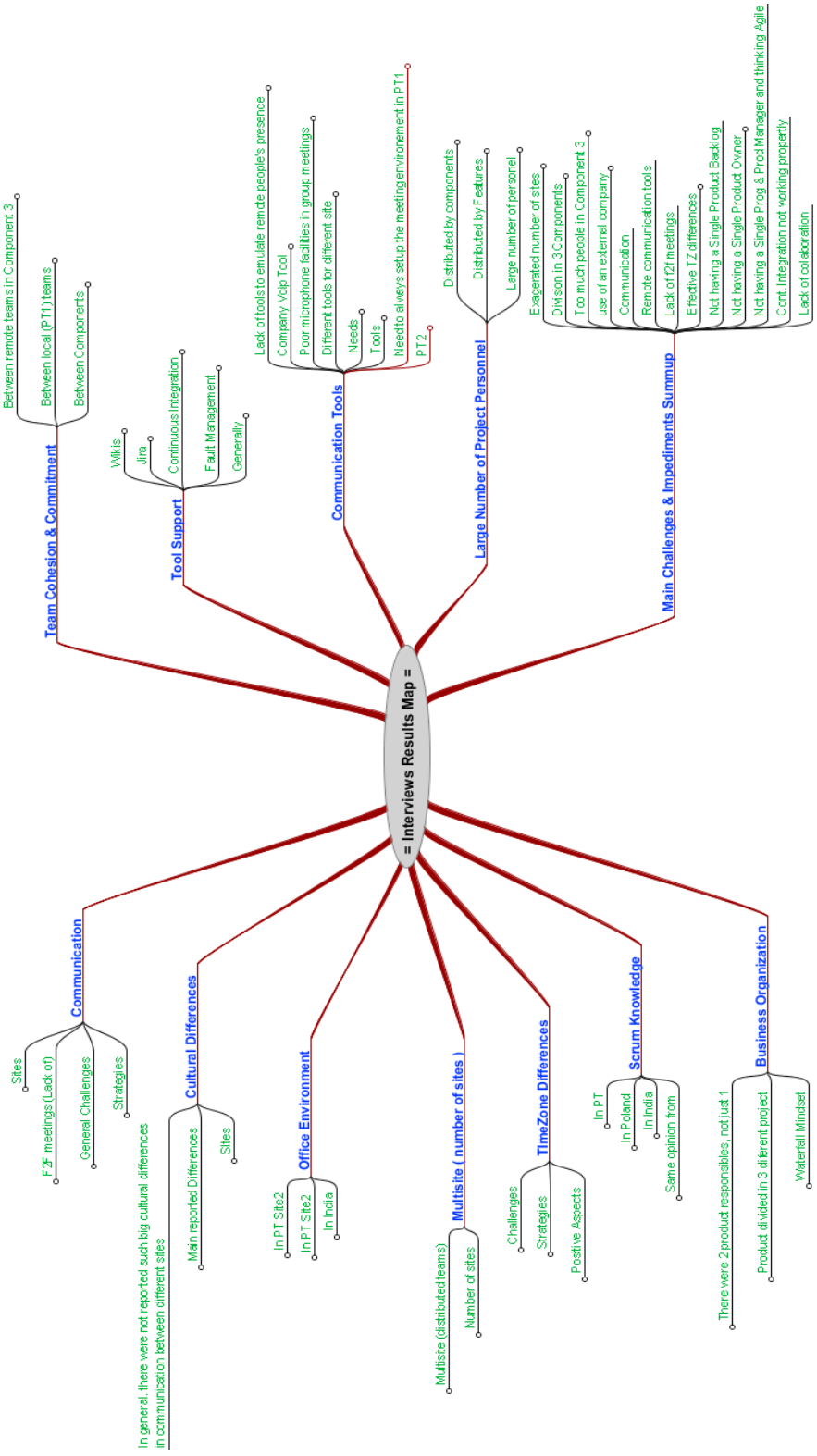
Scrum Master
1. Asynchronous
1.1 Can you list the main difficulties in the daily basis work that your teams are facing regarding time-zone differences?
1.2 Which are the practices applied in your teams to reduce the impact of time-zone differences in the daily work?
2. Lack of Group Awareness
2.1 Which are the communication challenges that your teams face with other remote Scrum teams? How does the communication challenges differ regarding the different remote sites?
2.2 Do your teams also face communication challenges with local Scrum teams? If yes, which are the main challenges?
2.3 How do you feel team cohesion and commitment between your teams and the multiple remote teams?
2.4 How do you feel the overall teams knowledge of Scrum practices and values? Is it an impediment for the overall process efficiency?
2.5 Which are the most significant cultural differences that you and your teams face when working with remote colleagues?
3. Poor communication Bandwidth
3.1 Which are the most significant impediments in your and yours' teams remote communication related with the available communication tools?
4. Lack of Tool Support
4.1 How do you think the available tools (Wikis, Jira, Cont. Integration, …) are efficient and enough to support the distributed Scrum practices?
4.2 In your opinion, people correctly use the available tools for the distributed Scrum practices? What do you advise to enhance its use?
5. Large Number of Project Personnel
5.1 Regarding your current experience, what are the main difficulties that the current distributed Scrum team assignment by components raise in yours' teams daily work?
5.2 How do you think the current distributed Scrum team assignment by components facilitates the communication between remote team members?
5.3 In your opinion, how should distributed Scrum teams be assigned, according to the actual product software architecture, in order to increase teams productivity?
6. Lack of Collaborative Office Environment
6.1 Which are the most significant impediments you and your team face with your office environment regarding Scrum practices?
7. Increased Number of Sites
7.1 Regarding your experience what are the positive and negative aspects of having the project distributed over multiple sites?
7.2 Do you think the current number of sites involved in the project is efficient? Should the number of sites be increased? Or decreased?
8. Open questions:
8.1 Regarding your experience can you describe the main difficulties in the project that impacts negatively the development productivity and product quality?
8.2 Which recommendations do you propose to overcome the above mentioned issues?

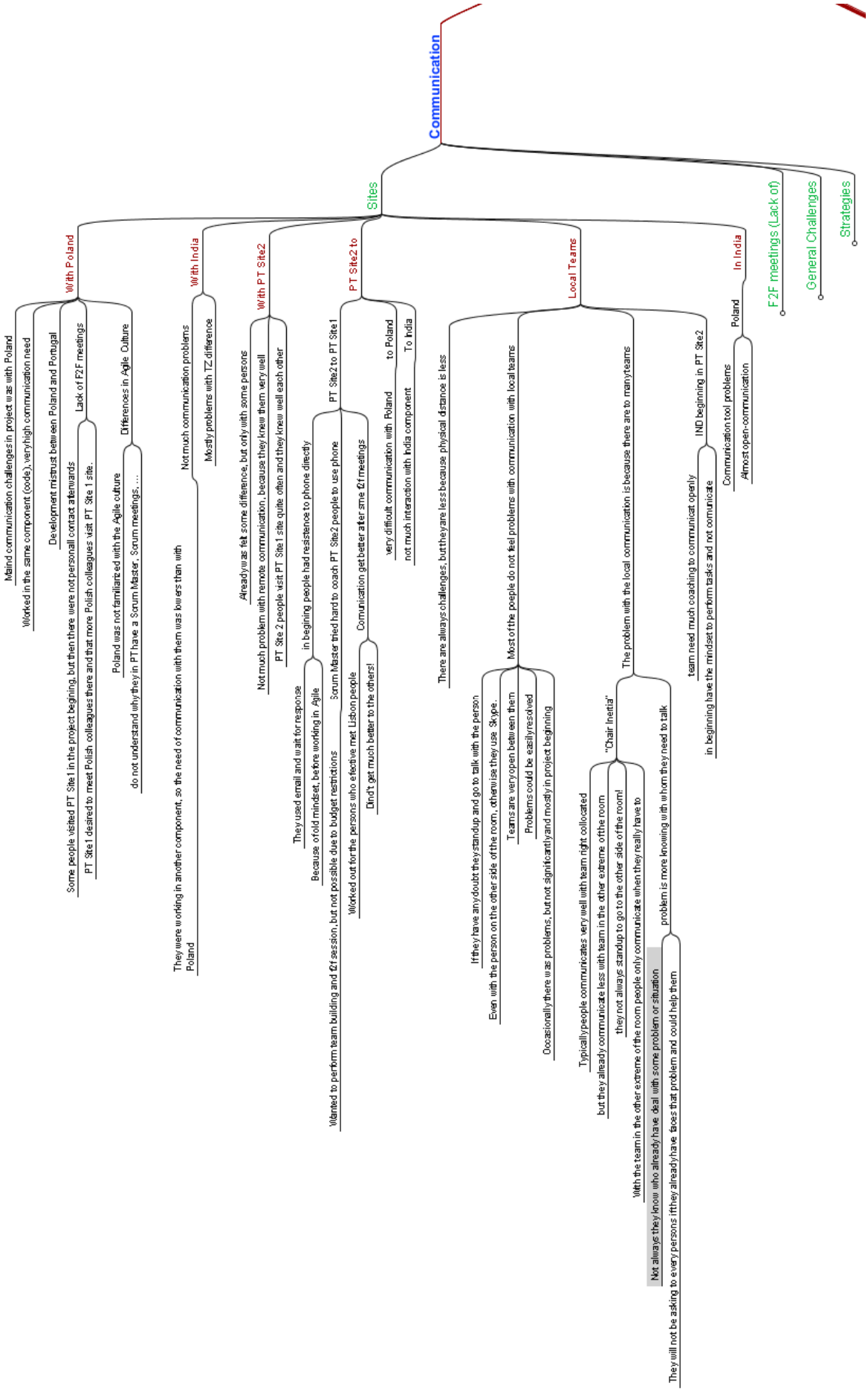
Product Owner
1. Asynchronous
1.1 Can you list the main difficulties that you face in your daily basis work regarding time-zone differences?
2. Lack of Group Awareness
2.1 As Product Owner how do you feel the overall distributed teams cohesion and commitment to achieve the best product quality in the end of each Sprint?
2.2 How frequently do you visit the other project's remote sites and teams?
2.3 Which are the main impediments you face in communicating customer requirements to the distributed Scrum teams?
3. Poor communication Bandwidth
3.1 Which are the most significant impediments in your remote communication with all teams related with the available communication tools?
4. Lack of Tool Support
4.1 How do you think the available tools (Wikis, Jira, Cont. Integration, …) are efficient and enough to support the distributed Scrum practices?
5. Large Number of Project Personnel
5.1 Regarding your experience in the project how the current distributed Scrum team assignment by components impacts the product quality and customer satisfaction?
5.2 In your opinion, how should distributed Scrum teams be assigned, according to the actual product software architecture, in order to increase teams productivity and overall product quality?
6. Lack of Collaborative Office Environment
6.1 Which are the most significant impediments you face with your office environment regarding Scrum practices?
7. Increased Number of Sites
7.1 Regarding your experience, the current number of distributed sites in the project is efficient and contributes to enhance product quality and customer satisfaction?
8. Open questions:
8.1 Regarding your experience can you describe the main difficulties in the project that impacts negatively the product quality and customer satisfaction?
8.2 Which recommendations do you propose to overcome the above mentioned issues?

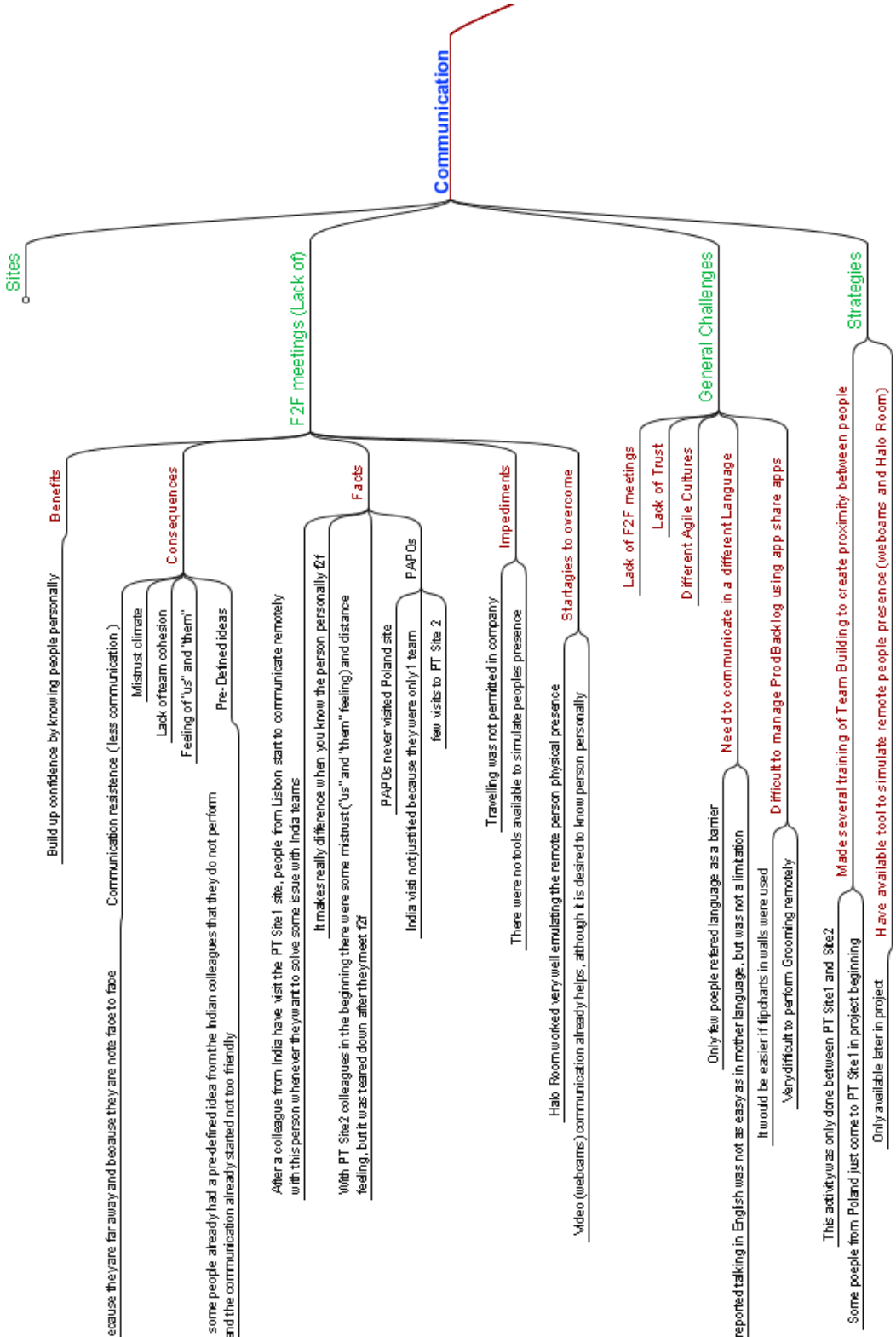
Manager
1. Asynchronous
1.1 What have been the measures applied in the project to address difficulties regarding time-zone differences between the different project sites?
1.2 In what extend these difficulties with time-zone differences have impact in team communication and productivity?
2. Lack of Group Awareness
2.1 How, in this project, is the overall distributed Scrum teams commitment and cohesion to provide a quality product and meet customer expectations?
2.2 What are the practices used in the project to increase the distributed teams cohesion and commitment?
2.3 How often face-to-face meetings between some distributed Scrum team members are organized in the project?
3. Poor communication Bandwidth
3.1 Which are the most significant impediments in your remote communication with all teams related with the available communication tools?
4. Lack of Tool Support
4.1 How do you think the available tools (Wikis, Jira, Cont. Integration, …) are efficient and enough to support the distributed Scrum practices in the project?
5. Large Number of Project Personnel
5.1 What have been the benefits and disbenefits of the current distributed Scrum team assignment by components in the project?
5.3 In your opinion, how should distributed Scrum teams be assigned in order to increase projects' efficiency and final customer satisfaction?
6. Lack of Collaborative Office Environment
6.1 Which are the most significant impediments faced by Scrum teams with office environment regarding Scrum practices?
7. Increased Number of Sites
7.1 The current number of sites involved in the project is efficient in terms of costs, productivity and customer satisfaction?
7.2 The project and the final customer would benefit if the number of sites and distributed teams are incremented or decremented?
8. Open Questions:
8.1 Regarding the project background can you describe the main difficulties in the project that impacts negatively the development efficiency and final product quality?
8.2 Which measures being planned to overcome the above mentioned issues?

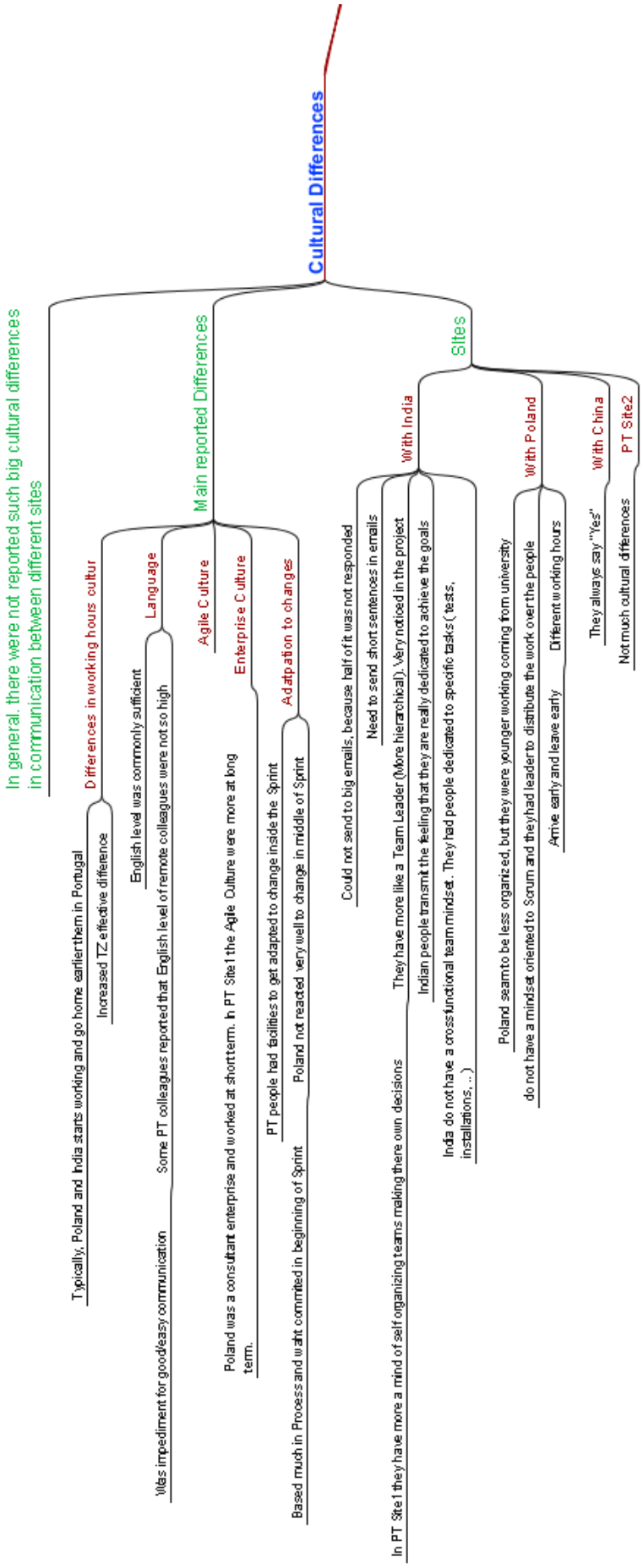
Site	Role	Date	Duration (min)	
PT1	ScrumMaster	13-Apr-11	57	
PT1	Team	19-Apr-11	49	
PT1	Team	19-Apr-11	53	
PT1	Team	20-Apr-11	48	
PT1	Team	21-Apr-11	59	
PT1	Manager	06-May-11	41	
PT1	PAPO	09-May-11	69	
PT1	PAPO	10-May-11	82	
PT2	ScrumMaster	16-May-11	52	
PT2	Team	18-May-11	48	
India	ScrumMaster	20-May-11	56	
		TOTALs	**614**	Minutes
			10	Hours
			11	Interviews

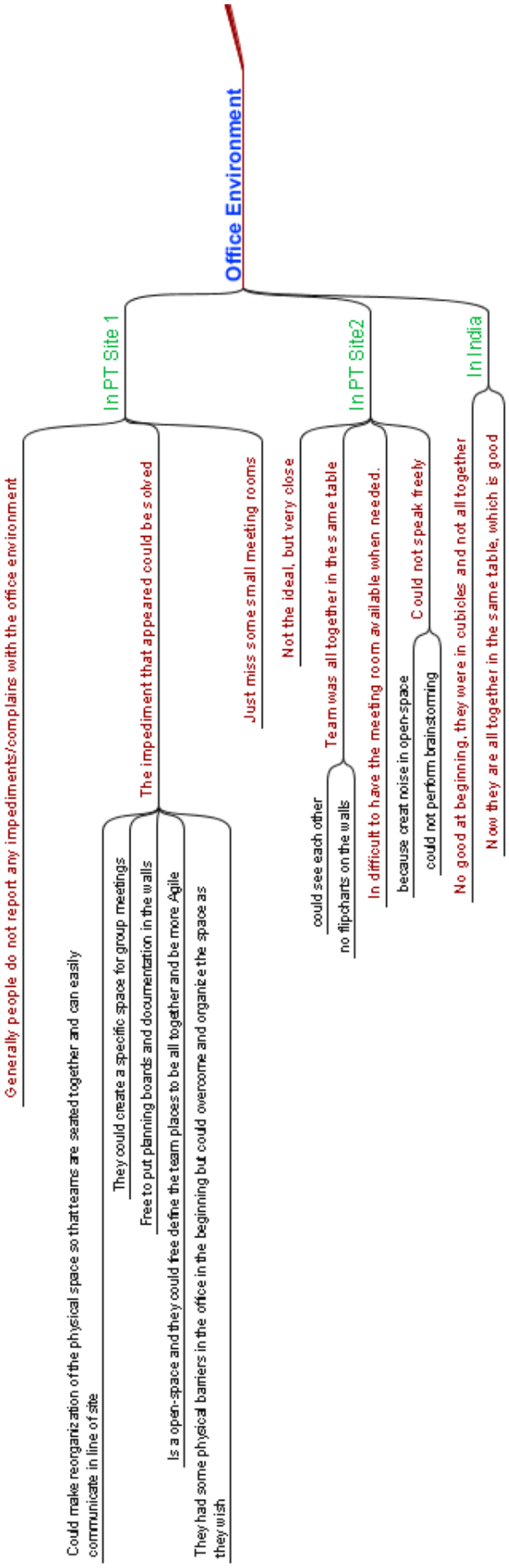
Annex B – Interviews Result Maps

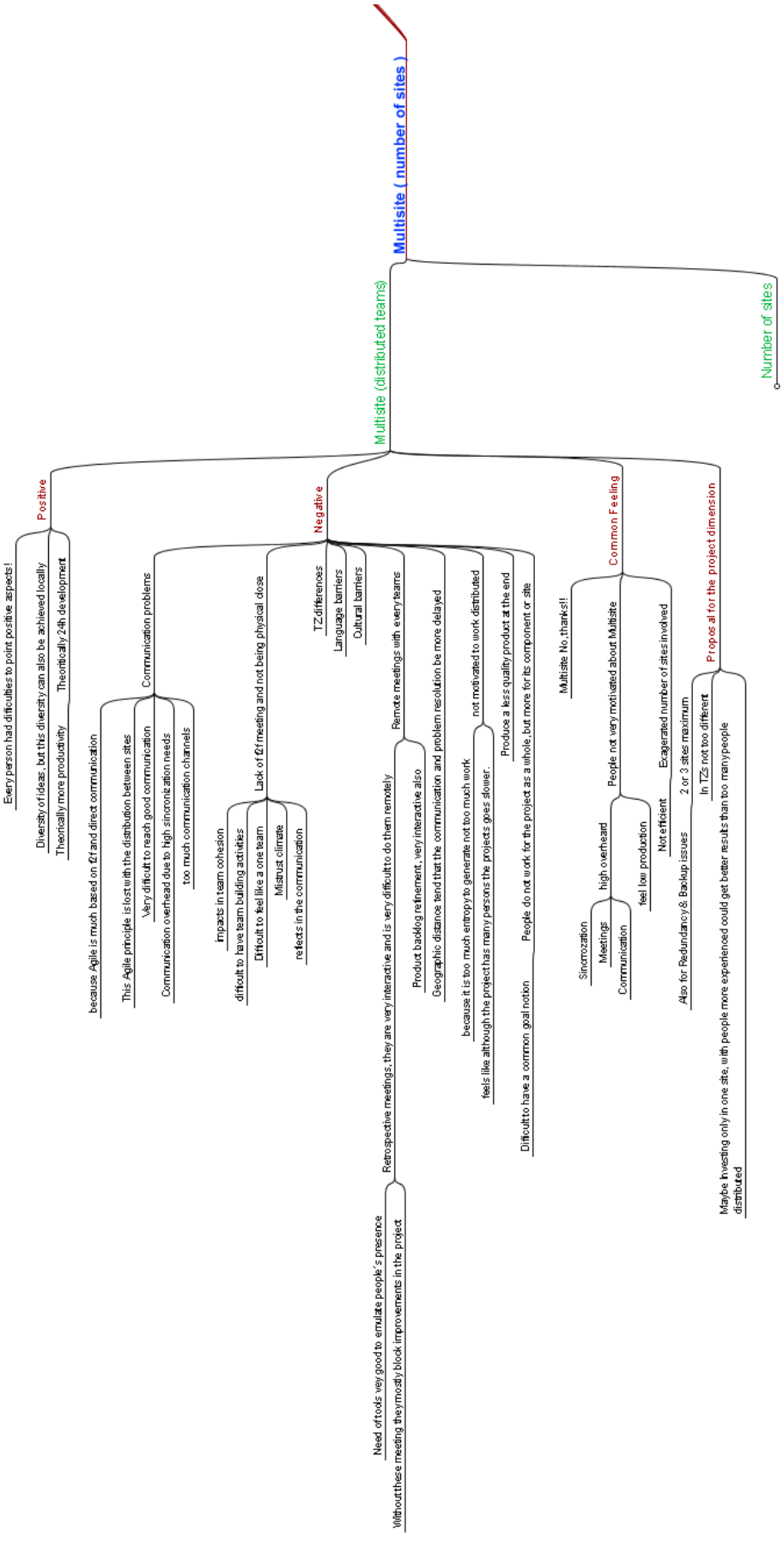


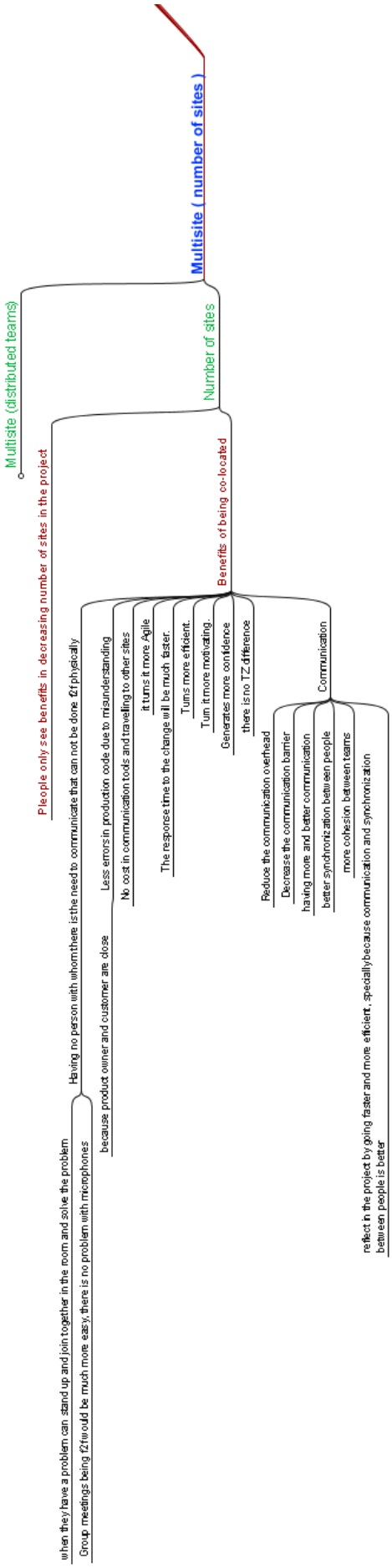

Communication
Sites
With Poland
Maind communication challenges in project was with Poland
Worked in the same component (code), veryhigh communication need
Development mistrust between Poland and Portugal
Some people visited PT Site1 in the project begining, but then there were not personall contact afterwards
PT Site1 desired to meet Polish colleagues there and that more Polish colleagues visit PT Site 1 site.
Lack of F2F meetings
Poland was not familiarized with the Agile culture
do not understand why they in PT have a Scrum Master, Scrum meetings, ...
Differences in Agile Culture
With India
They were working in another component, so the need of communication with them was lowers than with Poland
Not much communication problems
Mostly problems with TZ difference
With PT Site2
Already was felt some difference, but only with some persons
Not much problem with remote communication, because they knew them very well
PT Site 2 people visit PT Site1 site quite often and they knew well each other
PT Site2 to PT Site1
They used email and wait for response
Because of old mindset, before working in Agile
in beginning people had resistence to phone directly
Wanted to perform team building and f2f session, but not possible due to budget restrictions
Scrum Master tried hard to coach PT Site2 people to use phone
PT Site2 to PT Site1
Worked out for the persons who efective met Lisbon people
Comunication get better after sme f2f meetings
Dind't get much better to the others!
Local Teams
There are always challenges, but they are less because physical distance is less
very difficult communication with Poland
to Poland
not much interaction with India component
To India
If they have anydoubt they standup and go to talk with the person
Even with the person on the other side of the room, otherwise they use Skype.
Most of the poeple do not feel problems with communication with local teams
Teams are very open between them
Occasionally there was problems, but not significantly and mostly in project beginning
Problems could be easily resolved
Typically people communicates very well with team right collocated
but they already communicate less with team in the other extreme of the room
"Chair Inertia"
they not always standup to go to the other side of the room!
The problem with the local communication is because there are to many teams
With the team in the other extreme of the room people only communicate when they really have to
Not always they know who already have deal with some problem or situation
problem is more knowing with whom they need to talk
They will not be asking to every persons if they already have faces that problem and could help them
In India
team need much coaching to communicat openly
IND beginning in PT Site2
in beginning have the mindset to perform tasks and not comunicate
Communication tool problems
Poland
Almost open-communication
F2F meetings (Lack of)
General Challenges
Strategies

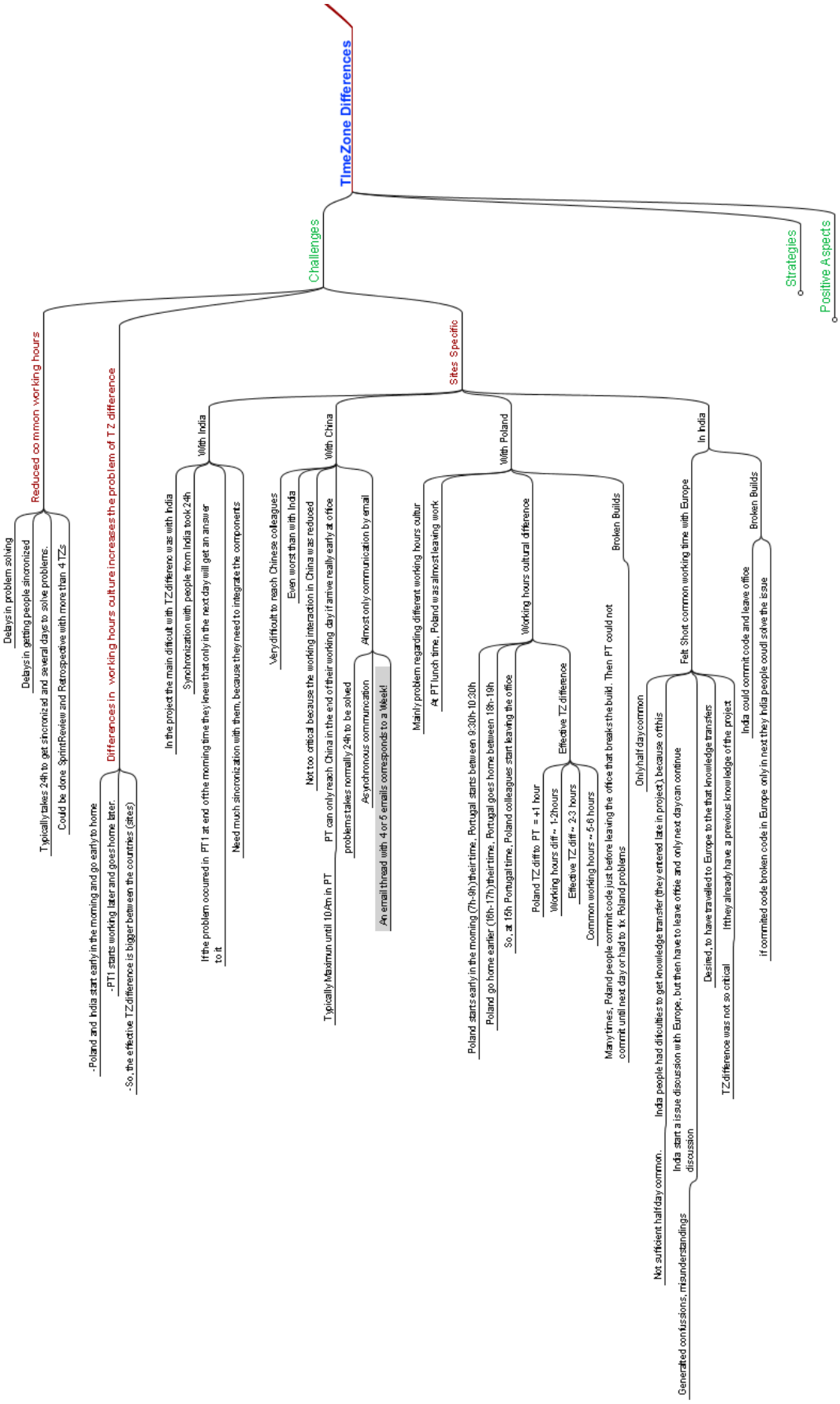

Communication
Sites
F2F meetings (Lack of)
General Challenges
Strategies
Benefits
Consequences
Facts
Impediments
Startagies to overcome
Lack of F2F meetings
Lack of Trust
Different Agile Cultures
Need to communicate in a different Language
Difficult to manage ProdBacklog using app share apps
Made several training of Team Building to create proximity between people
Have available tool to simulate remote people presence (webcams and Halo Room)
Build up confidence by knowing people personally
Communication resistence (less communication)
Mistrust climate
Lack of team cohesion
Feeling of "us" and "them"
Pre-Defined ideas
People communicate less because they are far away and because they are note face to face
In PT Site1, some people already had a pre-defined idea from the Indian colleagues that they do not perform a good job and the communication already started not too friendly
After a colleague from India have visit the PT Site1 site, people from Lisbon start to communicate remotely with this person whenever they want to solve some issue with India teams
It makes really difference when you know the person personally f2f
With PT Site2 colleagues in the beginning there were some mistrust ("us" and "them" feeling) and distance feeling, but it was teared down after they meet f2f
PAPOs never visited Poland site
India visit not justified because they were only 1 team
PAPOs
few visits to PT Site 2
Travelling was not permitted in company
There were no tools available to simulate peoples presence
Halo Room worked very well emulating the remote person physical presence
Video (webcams) communication already helps, although it is desired to know person personally
Only few poeple referred language as a barrier
most people reported talking in English was not as easy as in mother language, but was not a limitation
It would be easier if flipcharts in walls were used
Very difficult to perform Grooming remotely
This activity was only done between PT Site1 and Site2
Some poeple from Poland just come to PT Site1 in project beginning
Only available later in project

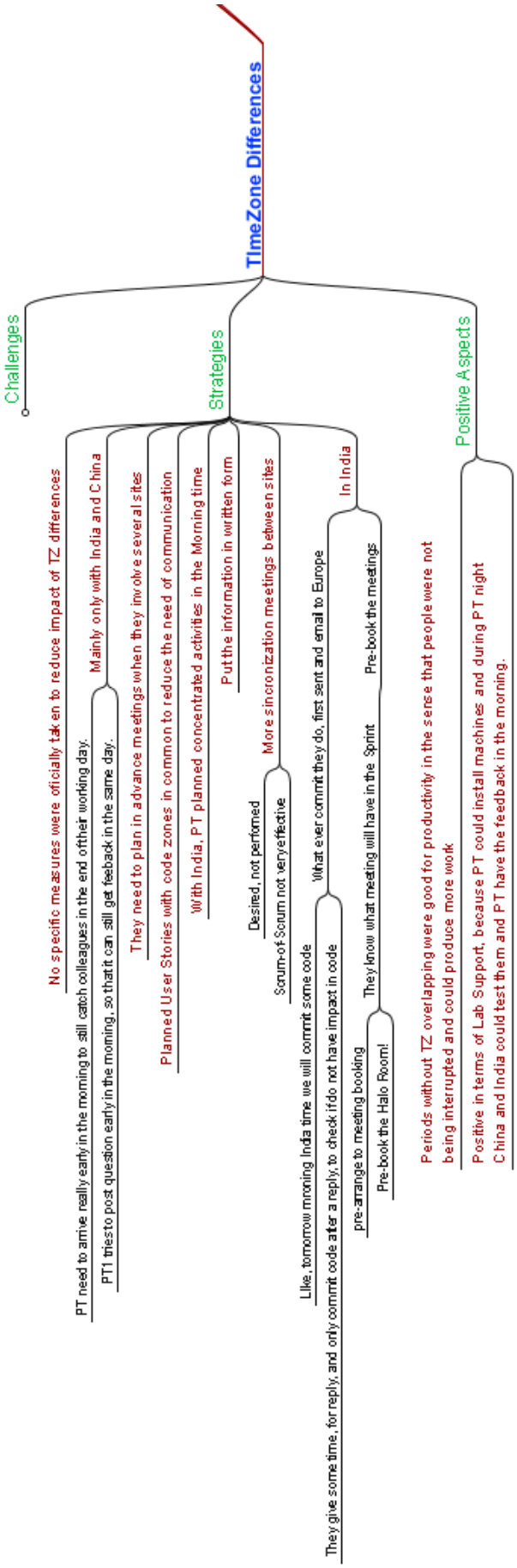

Cultural Differences
In general, there were not reported such big cultural differences in communication between different sites
Main reported Differences
Differences in working hours cultur
Typically, Poland and India starts working and go home earlier them in Portugal
Increased TZ effective difference
Language
English level was commonly sufficient
Some PT colleagues reported that English level of remote colleagues were not so high
Was impediment for good/easy communication
Agile Culture
Poland was a consultant enterprise and worked at short term. In PT Site1 the Agile Culture were more at long term.
Enterprise Culture
PT people had facilities to get adapted to change inside the Sprint
Adaptation to changes
Based much in Process and want commited in beginning of Sprint
Poland not reacted very well to change in middle of Sprint
Sites
With India
Could not send to big emails, because half of it was not responded
Need to send short sentences in emails
They have more like a Team Leader (More hierarchical). Very noticed in the project
In PT Site1 they have more a mind of self organizing teams making there own decisions
Indian people transmit the feeling that they are really dedicated to achieve the goals
India do not have a cross functional team mindset. They had people dedicated to specific tasks (tests, installations, ...)
With Poland
Poland seam to be less organized, but they were younger working coming from university
do not have a mindset oriented to Scrum and they had leader to distribute the work over the people
Arrive early and leave early
Different working hours
With China
They always say "Yes"
PT Site2
Not much cultural differences

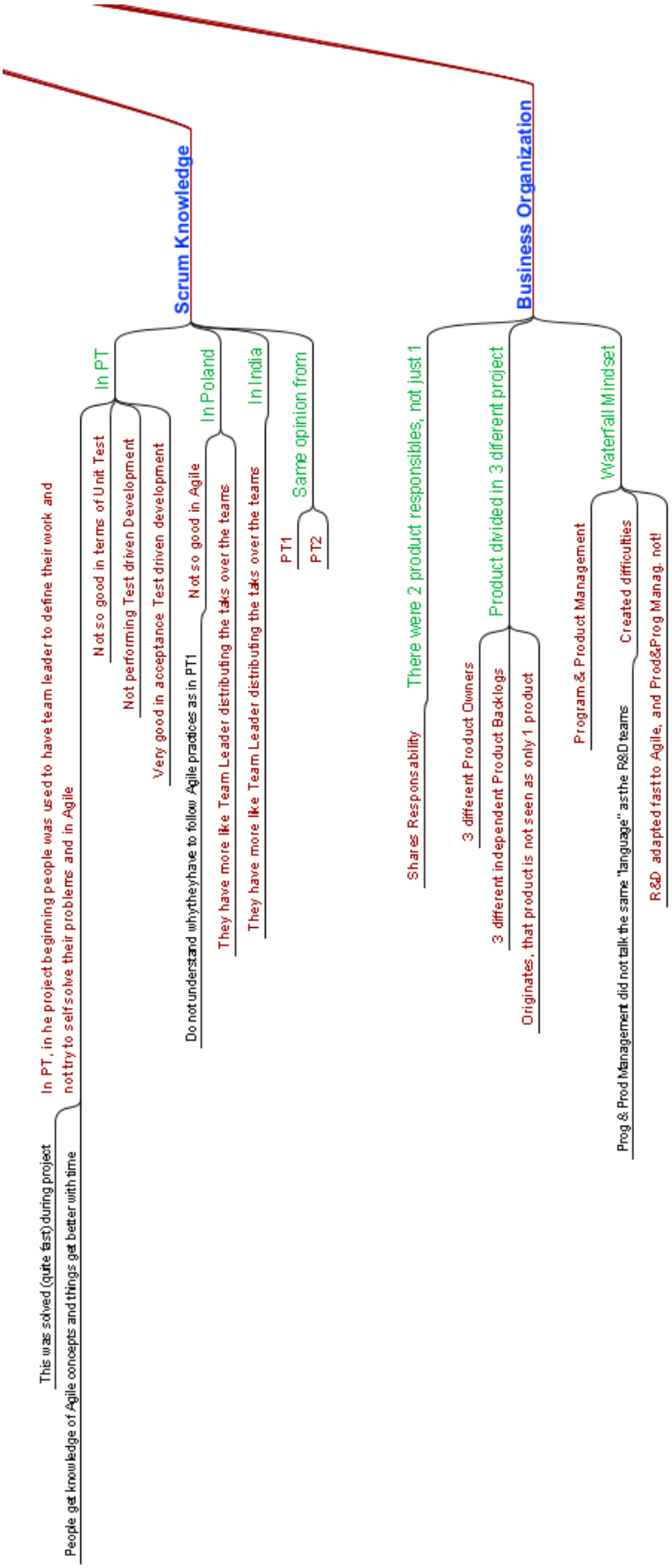

Office Environment
In PT Site 1
Generally people do not report any impediments/complains with the office environment
Could make reorganization of the physical space so that teams are seated together and can easily communicate in line of site
The impediment that appeared could be solved
They could create a specific space for group meetings
Free to put planning boards and documentation in the walls
Is a open-space and they could free define the team places to be all together and be more Agile
They had some physical barriers in the office in the beginning but could overcome and organize the space as they wish
Just miss some small meeting rooms
In PT Site2
Not the ideal, but very close
Team was all together in the same table
could see each other
no flipcharts on the walls
In difficult to have the meeting room available when needed.
because creat noise in open-space
Could not speak freely
could not perform brainstorming
In India
No good at beginning, they were in cubicles and not all together
Now they are all together in the same table, which is good

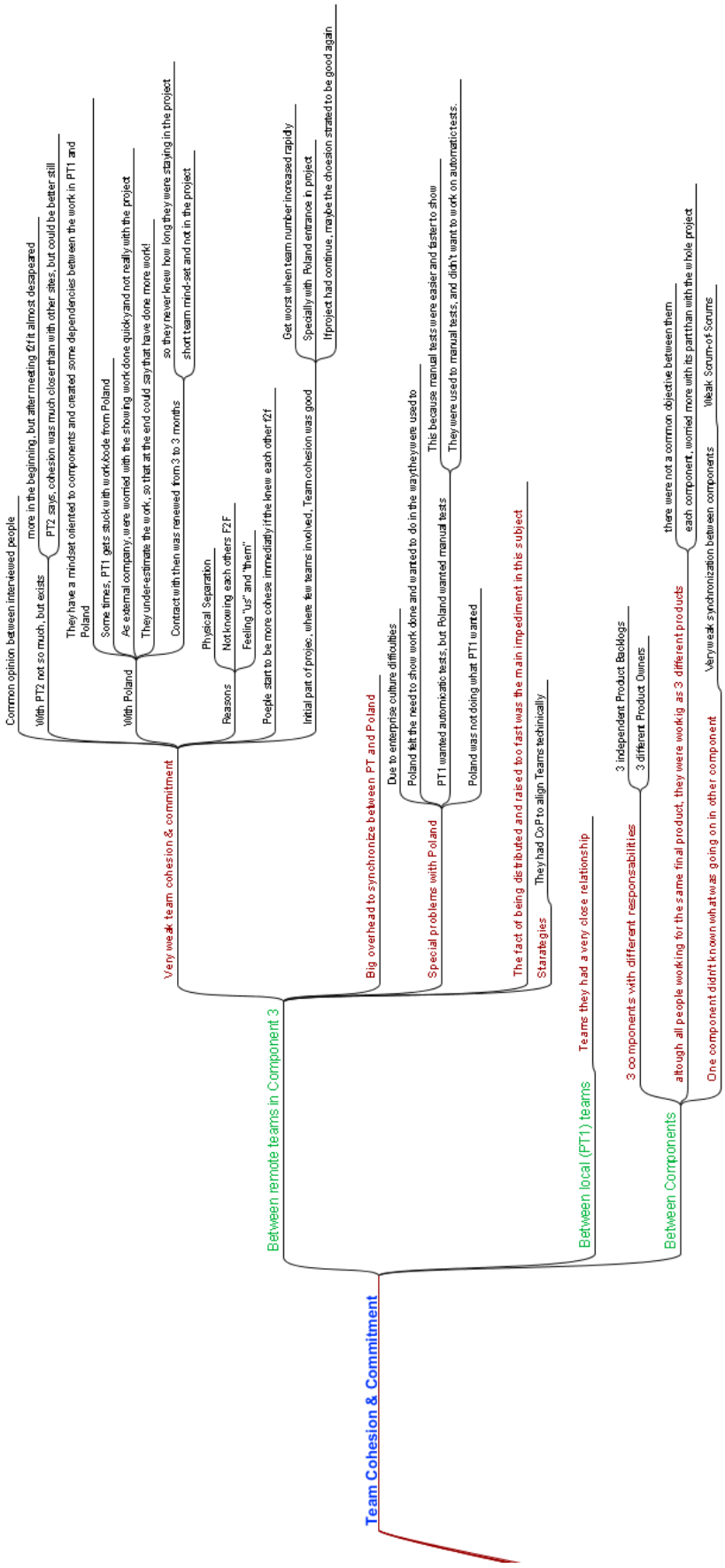

Multisite (number of sites)
Multisite (distributed teams)
Number of sites
Positive
Every person had difficulties to point positive aspects!
Diversity of ideas, but this diversity can also be achieved locally
Theorically more productivity
Theorically 24h development
because Agile is much based on f2f and direct communication
Negative
Communication problems
This Agile principle is lost with the distribution between sites
Very difficult to reach good communication
Communication overhead due to high sincronization needs
too much communication channels
Lack of f2f meeting and not being physical close
impacts in team cohesion
difficult to have team building activities
Difficult to feel like a one team
Mistrust climate
reflects in the communication
TZ differences
Language barriers
Cultural barriers
Remote meetings with every teams
Retrospective meetings, they are very interactive and is very difficult to do them remotely
Need of tools very good to emulate people's presence
Without these meeting they mostly block improvements in the project
Product backlog refinement, very interactive also
Geographic distance tend that the communication and problem resolution be more delayed
not motivated to work distributed
because it is too much entropy to generate not too much work
feels like although the project has many persons the projects goes slower.
Produce a less quality product at the end
Difficult to have a common goal notion
People do not work for the project as a whole, but more for its component or site
Common Feeling
Multisite No, thanks!!
People not very motivated about Multisite
high overhead
Meetings
Sincronization
Communication
feel low production
Not efficient
Exagerated number of sites involved
Also for Redundancy& Backup issues
2 or 3 sites maximum
In TZs not too different
Proposal for the project dimension
Maybe investing only in one site, with people more experienced could get better results than too many people distributed

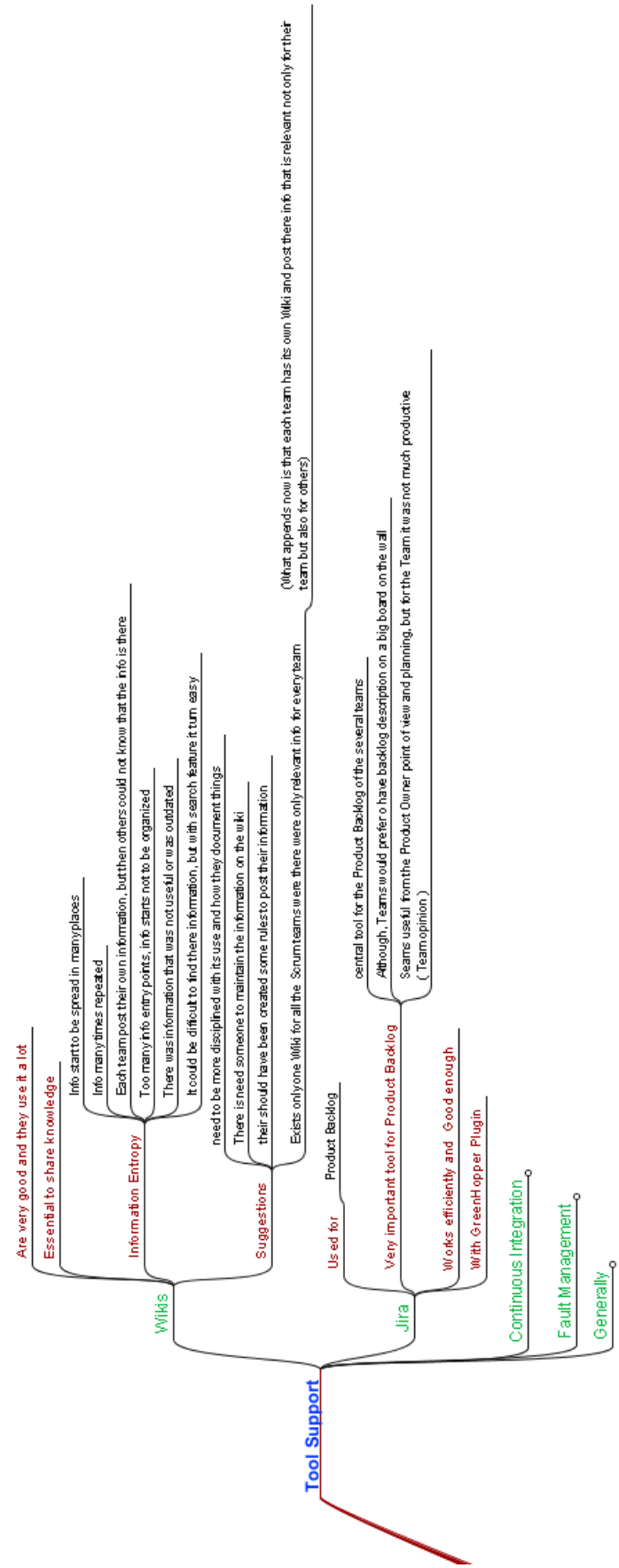

Multisite (number of sites)
Multisite (distributed teams)
Number of sites
Pleople only see benefits in decreasing number of sites in the project
Benefits of being co-located
Having no person with whom there is the need to communicate that can not be done f2f physically
when they have a problem can stand up and join together in the room and solve the problem
Group meetings being f2f would be much more easy, there is no problem with microphones
Less errors in production code due to misunderstanding
because product owner and customer are close
No cost in communication tools and travelling to other sites
it turns it more Agile
The response time to the change will be much faster.
Turns more efficient.
Turn it more motivating.
Generates more confidence
there is no TZ difference
Communication
Reduce the communication overhead
Decrease the communication barrier
having more and better communication
better synchronization between people
more cohesion between teams
reflect in the project by going faster and more efficient, specially because communication and synchronization
between people is better


TimeZone Differences
Challenges
Strategies
Positive Aspects
Reduced common working hours
Delays in problem solving
Delays in getting people sincronized
Typically takes 24h to get sincronized and several days to solve problems.
Could be done SprintReview and Retrospective with more than 4 TZs
Differences in working hours culture increases the problem of TZ difference
-Poland and India start early in the morning and go early to home
-PT1 starts working later and goes home later.
-So, the effective TZ difference is bigger between the countries (sites)
Sites Specific
With India
In the project the main difficult with TZ difference was with India
Synchronization with people from India took 24h
If the problem occurred in PT1 at end of the morning time they knew that only in the next day will get an answer to it
Need much sincronization with them, because they need to integrate the components
With China
Very difficult to reach Chinese colleagues
Even worst than with India
Not too critical because the working interaction in China was reduced
Typically Maximun until 10Am in PT
PT can only reach China in the end of their working day if arrive really early at office
problems takes normally 24h to be solved
Asynchronous communication
Almost only communication by email
An email thread with 4 or 5 emails corresponds to a Week!
With Poland
Mainly problem regarding different working hours cultur
At PT lunch time, Poland was almost leaving work
Poland starts early in the morning (7h-9h) their time, Portugal starts between 8:30h-10:30h
Poland go home earlier (16h-17h) their time, Portugal goes home between 18h-19h
So, at 16h Portugal time, Poland colleagues start leaving the office
Working hours cultural difference
Poland TZ diff to PT = +1 hour
Working hours diff ~ 1-2hours
Effective TZ difference
Effective TZ diff ~ 2-3 hours
Common working hours ~ 5-6 hours
Many times, Poland people commit code just before leaving the office that breaks the build. Then PT could not commit until next day or had to fix Poland problems
Broken Builds
Only half day common
Not sufficient half day common.
India people had dificulties to get knowledge transfer (they entered late in project), because of this
Generated confussions, misunderstandings
India start a issue discussion with Europe, but then have to leave offsite and only next day can continue discussion
In India
Felt Short common working time with Europe
Desired, to have travelled to Europe to the that knowledge transfers
TZ difference was not so critical
If they already have a previous knowledge of the project
India could commit code and leave office
Broken Builds
if commited code broken code in Europe, only in next they India people could solve the issue

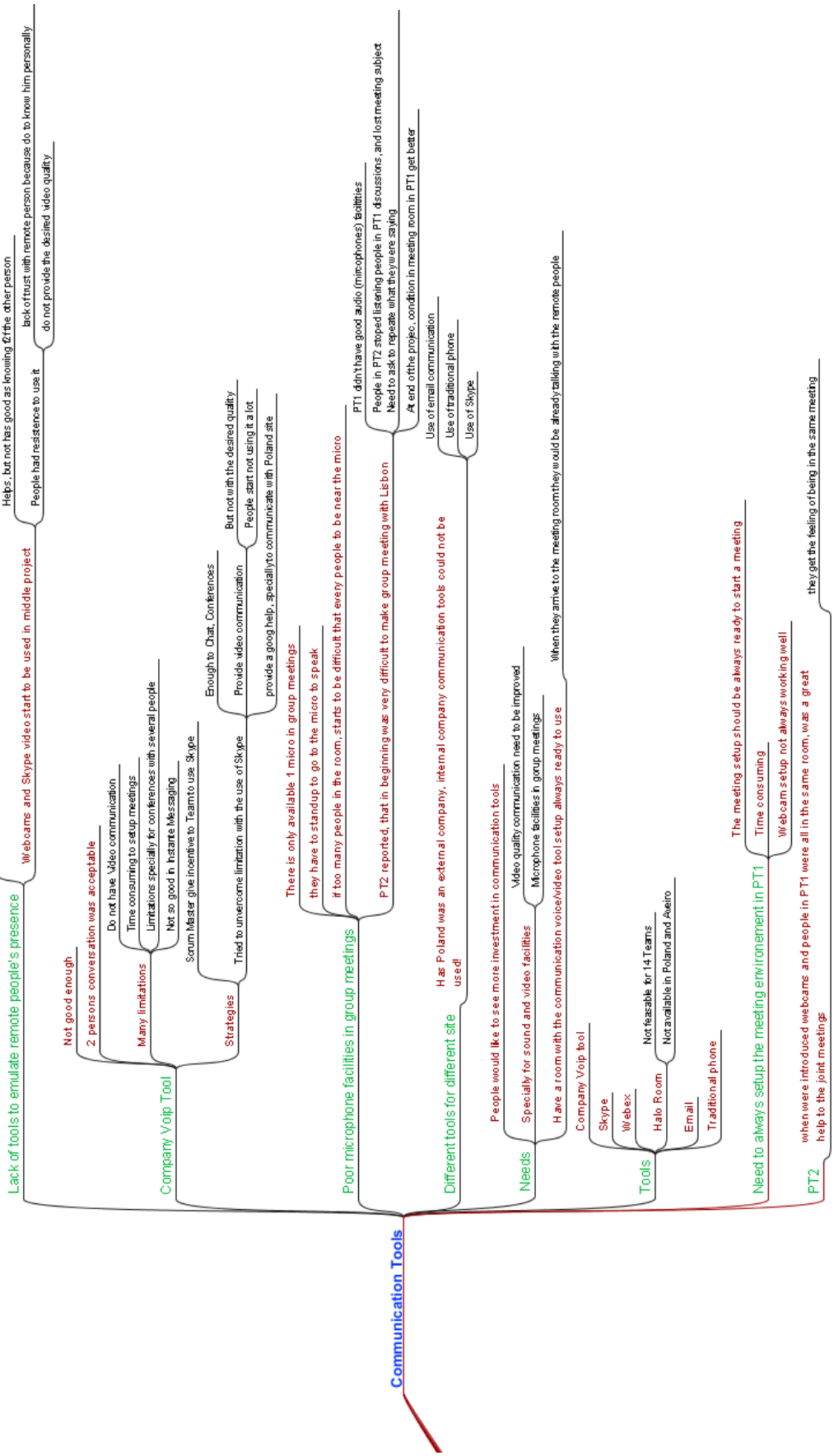

TImeZone Differences
Challenges
Strategies
Positive Aspects
No specific measures were officially taken to reduce impact of TZ differences
PT need to arrive really early in the morning to still catch colleagues in the end of their working day.
PT1 tries to post question early in the morning, so that it can still get feeback in the same day.
Mainly only with India and China
They need to plan in advance meetings when they involve several sites
Planned User Stories with code zones in common to reduce the need of communication
With India, PT planned concentrated activities in the Morning time
Put the information in written form
Desired, not performed
Scrum-of-Scrum not very effective
More sincronization meetings between sites
Like, tomorrow morning India time we will commit some code
They give some time, for reply, and only commit code after a reply, to check if do not have impact in code
What ever commit they do, first sent and email to Europe
In India
pre-arrange to meeting booking
Pre-book the Halo Room!
They know what meeting will have in the Sprint
Pre-book the meetings
Periods without TZ overlapping were good for productivity in the sense that people were not being interrupted and could produce more work
Positive in terms of Lab Support, because PT could install machines and during PT night China and India could test them and PT have the feedback in the morning.

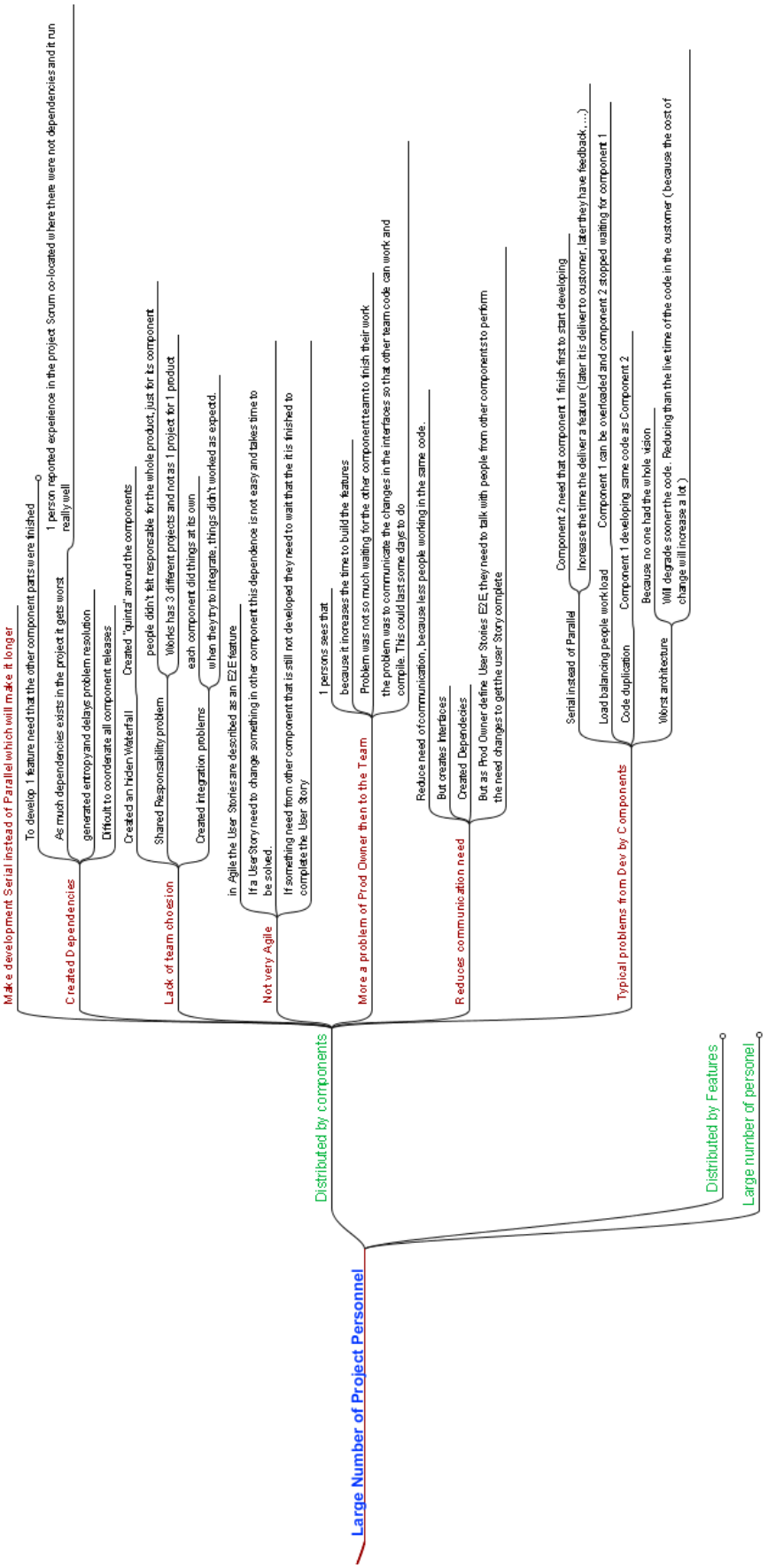

Scrum Knowledge
In PT
Not so good in terms of Unit Test
Not performing Test driven Development
Very good in acceptance Test driven development
In Poland
Not so good in Agile
Do not understand whytheyhave to follow Agile practices as in PT1
They have more like Team Leader distributing the taks over the teams
In India
They have more like Team Leader distributing the taks over the teams
Same opinion from
PT1
PT2
In PT, in he project beginning people was used to have team leader to define their work and nottry to selfsolve their problems and in Agile
This was solved (quite fast) during project
People get knowledge of Agile concepts and things get better with time
Business Organization
There were 2 product responsibles, not just 1
Shares Responsability
Product divided in 3 diferent project
3 different Product Owners
3 different independent Product Backlogs
Originates, that product is not seen as only 1 product
Waterfall Mindset
Program & Product Management
Created difficulties
Prog & Prod Management did not talk the same "language" as the R&D teams
R&D adapted fast to Agile, and Prod&Prog Manag. not!

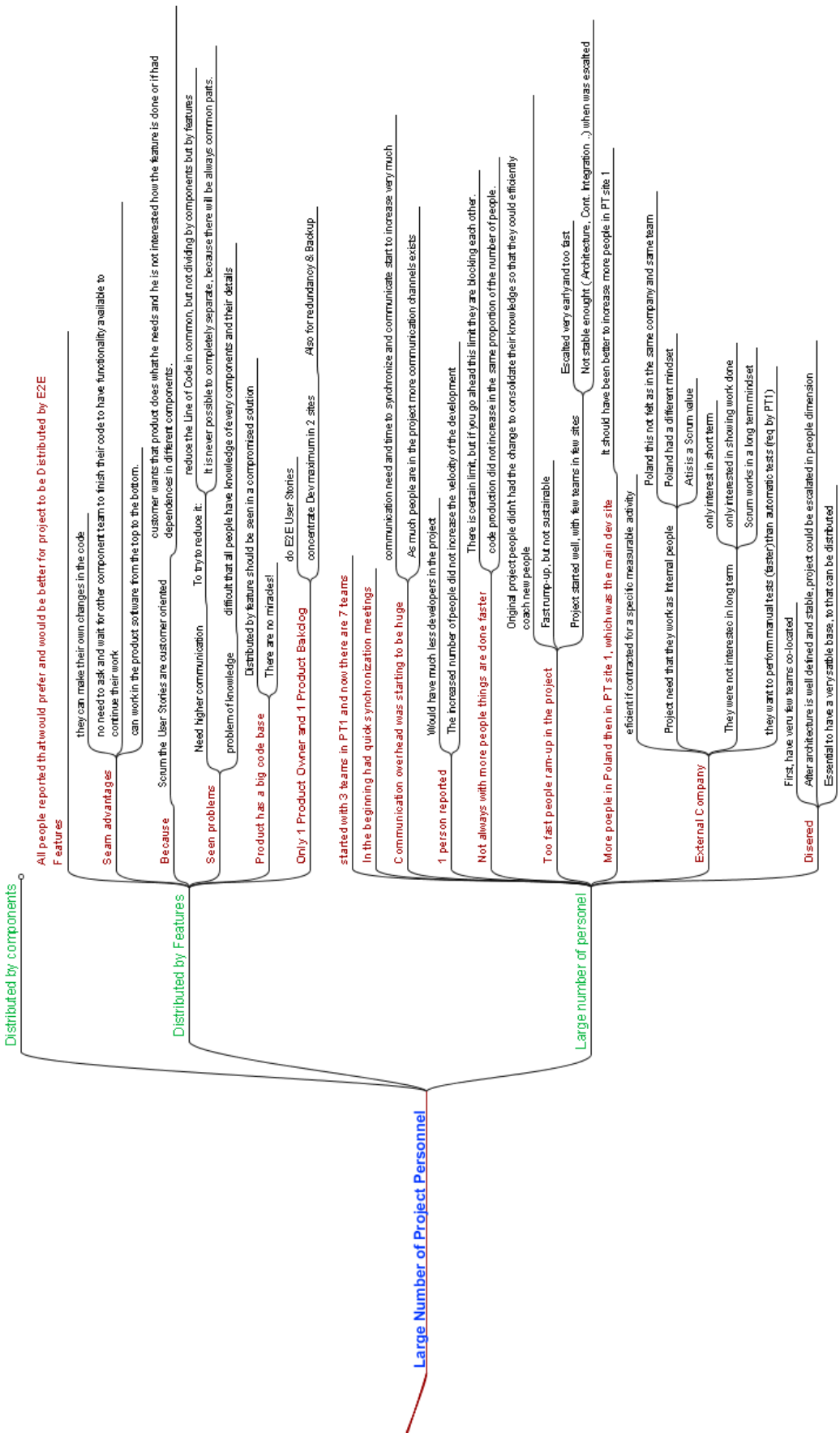

Team Cohesion & Commitment
Between remote teams in Component 3
Very weak team cohesion & commitment
Common opinion between interviewed people
With PT2 not so much, but exists
more in the beginning, but after meeting f2f it almost desapeared
PT2 says, cohesion was much closer than with other sites, but could be better still
They have a mindset oriented to components and created some dependencies between the work in PT1 and Poland
With Poland
Some times, PT1 gets stuck with work/code from Poland
As external company, were worried with the showing work done quick/yand not really with the project
They under-estimate the work, so that at the end could say that have done more work!
so they never knew how long they were staying in the project
Contract with then was renewed from 3 to 3 months
short team mind-set and not in the project
Reasons
Physical Separation
Not knowing each others F2F
Feeling "us" and "them"
People start to be more cohese immediatly if the knew each other f2f
Initial part of projec, where few teams involved. Team cohesion was good
Get worst when team number increased rapidly
Specially with Poland entrance in project
If project had continue, maybe the chohesion strated to be good again
Big overhead to synchronize between PT and Poland
Due to enterprise culture difficulties
Special problems with Poland
Poland felt the need to show work done and wanted to do in the way they were used to
PT1 wanted automicatic tests, but Poland wanted manual tests
This because manual tests were easier and faster to show
They were used to manual tests, and didn't want to work on automatic tests.
Poland was not doing what PT1 wanted
The fact of being distributed and raised too fast was the main impediment in this subject
Strategies
They had CoP to align Teams technically
Between local (PT1) teams
Teams they had a very close relationship
Between Components
3 components with different responsabilities
3 independent Product Backlogs
3 different Product Owners
although all people working for the same final product, they were workig as 3 different products
there were not a common objective between them
each component, worried more with its part than with the whole project
One component didn't known what was going on in other component
Very weak synchronization between components
Weak Scrum-of-Scrums

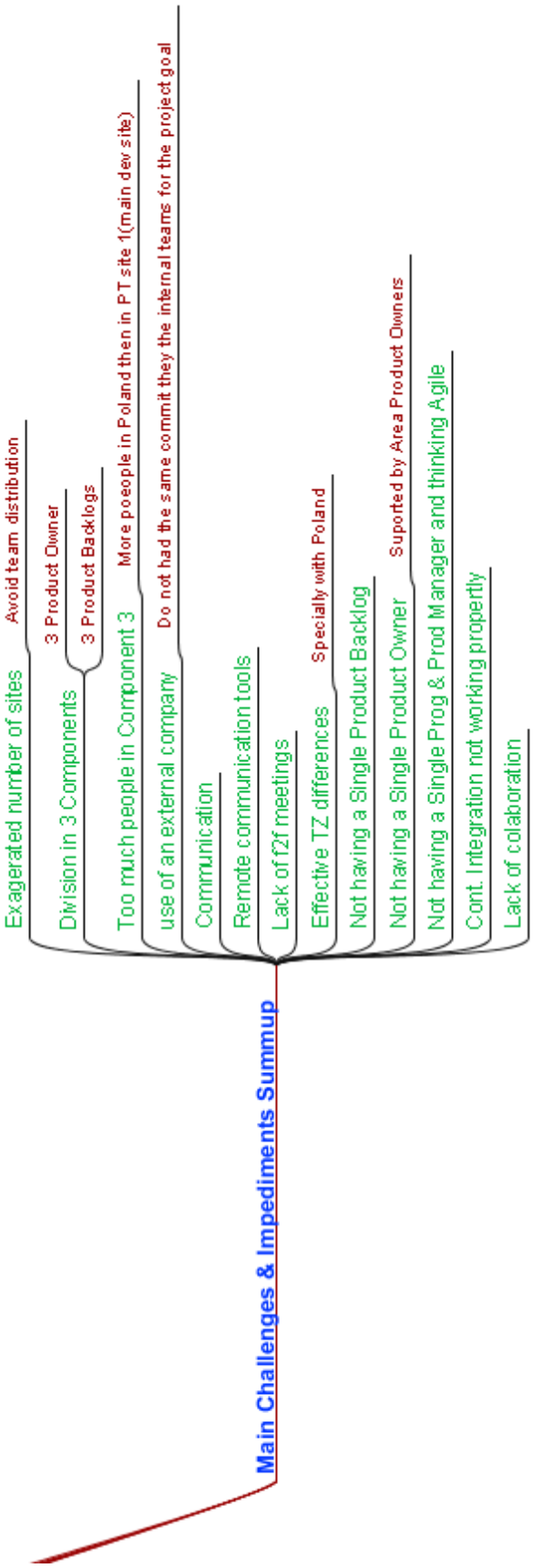

Tool Support
Wikis
Are very good and they use it a lot
Essential to share knowledge
Info start to be spread in manyplaces
Info many times repeated
Each team post their own information, but then others could not know that the info is there
Information Entropy
Too many info entry points, info starts not to be organized
There was information that was not useful or was outdated
It could be difficult to find there information, but with search feature it turn easy
need to be more disciplined with its use and how they document things
There is need someone to maintain the information on the wiki
their should have been created some rules to post their information
Suggestions
Exists only one Wiki for all the Scrum teams were there were only relevant info for every team
(What appends now is that each team has its own Wiki and post there info that is relevant not only for their team but also for others)
Jira
Used for Product Backlog
central tool for the Product Backlog of the several teams
Very important tool for Product Backlog
Although, Teams would prefer o have backlog description on a big board on the wall
Seams useful from the Product Owner point of view and planning, but for the Team it was not much productive
(Team opinion)
Works efficiently and Good enough
With GreenHopper Plugin
Continuous Integration
Fault Management
Generally


Tool Support
Wikis
Jira
Continuous Integration
Starts to work fine, but in the beginning there were many problems
Since start working fine is very good and efficient
Essential in distributed environment
Very productive to have it the project
Poland had difficulties to understand for what CI was needed for in the beginning
At the end of the project there were too many people working on it and Cont Inte efficiency decreased
Composition
Bamboo
Automatic Build Production
SubVersion
Version Control
Internal Company tool
Build compilation, installation in lab machine, run Smoke and Regression tests.
MPP
They need to open fault reports to fix things
Had some problems
Time needed to solve faults
Not very Agile
Test Automation
Robot
Had some problems for GUI tests
Need some time from Dev to get the tests running
many problems in the beginning to get it working fine, not because of the tool, but related with the test automation
Initial Problems
started automating test too late when the product already had many features
lack of clear rules how to develop tests (which libraries to use, common functions ...)
there were 1 test per User Story, which was too much
The number of tests to perform by User Story
no organization in the tests
from the Proxy Product Owner, there was not orientation to create a test per User Story, but by E2E functionality
they started to generate functional E2E tests to reduce the number of tests
Solution
modifying existing tests and not build up new ones
need to create a Taskforce team to re-organize and stabilize these parts of the automatic tests
Fault Management
Product bugtrack tool
it could be a easier tool to use
it has many information to fill that was not necessary
Generally
People think tool support is good enough for Scrum practices
People reported some problems in beginning of project, but now tools are working fine
Only could be better in Wikis information management
People usage of tools is generally good
people should committing more often their code
Cont. Integration usage
Some people were afraid of committing code and delayed process


Communication Tools
Lack of tools to emulate remote people's presence
Halo Room only available late in project and difficult to book it for meetings
very productive being able to see the remote people
Helps, but not has good as knowing if the other person
Webcams and Skype video start to be used in middle project
lack of trust with remote person because do to know him personally
People had resistence to use it
do not provide the desired video quality
Company Voip Tool
Not good enough
2 persons conversation was acceptable
Many limitations
Do not have Video communication
Time consuming to setup meetings
Limitations specially for conferences with several people
Not so good in Instante Messaging
Scrum Master give incentive to Team to use Skype
Strategies
Enough to Chat, Conferences
Tried to unvercome limitation with the use of Skype
But not with the desired quality
Provide video communication
People start not using it a lot
provide a goog help, specially to communicate with Poland site
Poor microphone facilities in group meetings
There is only available 1 micro in group meetings
they have to standup to go to the micro to speak
if too many people in the room, starts to be difficult that every people to be near the micro
PT2 reported, that in beginning was very difficult to make group meeting with Lisbon
PT1 didn't have good audio (microphones) facilities
People in, PT2 stoped listening people in PT1 discussions, and lost meeting subject
Need to ask to repeate what they were saying
At end of the projec, condition in meeting room in PT1 get better
Different tools for different site
Has Poland was an external company, internal company communication tools could not be used!
Use of email communication
Use of traditional phone
Use of Skype
Needs
People would like to see more investment in communication tools
Specially for sound and video facilities
Video quality communication need to be improved
Microphone facilities in group meetings
Have a room with the communication voice/video tool setup always ready to use
When they arrive to the meeting room they would be already talking with the remote people
Tools
Company Voip tool
Skype
Webex
Halo Room
Not feasable for 14 Teams
Not available in Poland and Aveiro
Email
Traditional phone
Need to always setup the meeting environement in PT1
The meeting setup should be always ready to start a meeting
Time consuming
Webcam setup not always working well
PT2
when were introduced webcams and people in PT1 were all in the same room, was a great help to the joint meetings
they get the feeling of being in the same meeting


Large Number of Project Personnel
Distributed by components
Distributed by Features
Large number of personel
Make development Serial instead of Parallel which will make it longer
To develop 1 feature need that the other component parts were finished
Created Dependencies
As much dependencies exists in the project it gets worst
1 person reported experience in the project Scrum co-located where there were not dependencies and it run really well
generated entropy and delays problem resolution
Difficult to coordenate all component releases
Lack of team chohesion
Created an hidden Waterfall
Created "quintal" around the components
Shared Responsability problem
people didn't felt responsable for the whole product, just for its component
Works has 3 different projects and not as 1 project for 1 product
Created integration problems
each component did things at its own
when they try to integrate, things didn't worked as expected.
Not very Agile
in Agile the User Stories are described as an E2E feature
If a User Story need to change something in other component this dependence is not easy and takes time to be solved.
If something need from other component that is still not developed they need to wait that the it is finished to complete the User Story
More a problem of Prod Owner then to the Team
1 persons sees that
because it increases the time to build the features
Problem was not so much waiting for the other component team to finish their work
the problem was to communicate the changes in the interfaces so that other team code can work and compile. This could last some days to do
Reduces communication need
Reduce need of communication, because less people working in the same code.
But creates Interfaces
Created Dependecies
But as Prod Owner define User Stories E2E they need to talk with people from other components to perform the need changes to get the user Story complete
Typical problems from Dev by Components
Serial instead of Parallel
Component 2 need that component 1 finish first to start developing
Increase the time the deliver a feature (later it is deliver to customer, later they have feedback....)
Load balancing people work load
Component 1 can be overloaded and component 2 stopped waiting for component 1
Code duplication
Component 1 developing same code as Component 2
Worst architecture
Because no one had the whole vision
Will degrade sooner the code. Reducing than the live time of the code in the customer (because the cost of change will increase a lot.)


Large Number of Project Personnel
Distributed by components
All people reported that would prefer and would be better for project to be Distributed by E2E Features
Distributed by Features
Seam advantages
they can make their own changes in the code
no need to ask and wait for other component team to finish their code to have functionality available to continue their work
can work in the product software from the top to the bottom.
Because
Scrum the User Stories are customer oriented
customer wants that product does what he needs and he is not interested how the feature is done or if had dependences in different components:.
Seen problems
Need higher communication
To try to reduce it:
reduce the Line of Code in common, but not dividing by components but by features
It is never possible to completely separate, because there will be always common parts.
problem of knowledge
difficult that all people have knowledge of every components and their details
Product has a big code base
Distributed by feature should be seen in a compromised solution
There are no miracles!
Only 1 Product Owner and 1 Product Bakclog
do E2E User Stories
concentrate Dev maximum in 2 sites
Also for redundancy & Backup
started with 3 teams in PT1 and now there are 7 teams
In the beginning had quick synchronization meetings
Communication overhead was starting to be huge
communication need and time to synchronize and communicate start to increase very much
As much people are in the project more communication channels exists
1 person reported
Would have much less developers in the project
The increased number of people did not increase the velocity of the development
Not always with more people things are done faster
There is certain limit, but if you go ahead this limit they are blocking each other.
code production did not increase in the same proportion of the number of people.
Original project people didn't had the change to consolidate their knowledge so that they could efficiently coach new people
Too fast people ram-up in the project
Fast rump-up, but not sustainable
Large number of personel
Project started well, with few teams in few sites
Escalted very early and too fast
Not stable enought (Architecture, Cont. Integration ..) when was escalted
More poeple in Poland then in PT site 1, which was the main dev site
It should have been better to increase more people in PT site 1
efficient if contracted for a specific measurable activity
External Company
Project need that they work as Internal people
Poland this not felt as in the same company and same team
Poland had a different mindset
They were not interestec in long term
Atisis a Scrum value
only interest in short term
only interested in showing work done
Scrum works in a long term mindset
they want to perform manual tests (faster) than automatic tests (req by PT1)
Disered
First, have veru few teams co-located
After architecture is well defined and stable, project could be escalated in people dimension
Essential to have a very stable base, to that can be distributed


Main Challenges & Impediments Summup
Exagerated number of sites
Avoid team distribution
Division in 3 Components
3 Product Owner
3 Product Backlogs
Too much people in Component 3
More poeople in Poland then in PT site 1(main dev site)
use of an external company
Do not had the same commit they the internal teams for the project goal
Communication
Remote communication tools
Lack of f2f meetings
Effective TZ differences
Specially with Poland
Not having a Single Product Backlog
Not having a Single Product Owner
Suported by Area Product Owners
Not having a Single Prog & Prod Manager and thinking Agile
Cont. Integration not working properly
Lack of colaboration

This page intentionally left blank

Annex C – Scrum practices details

Sprint Planning

Sprint Planning part 1

1. First people got in place and meeting facilities were prepared. PBL is made visible to every Team member in the meeting. It were clarified the Sprint events, like Sprint end date, national holidays in each of the sites, any trainings being taken, and so on. The Definition of Done (DoD) was also reviewed and agreed.
2. In the second step, PAPO explained the priorities for the Sprint to the Team representatives.
3. PAPO set the Sprint goals with the Team representatives
4. Teams choose USs they thought they could do during the Sprint and took them to Sprint Planning part 2 to be then analysed, break USs into tasks, estimate their effort and make the final Sprint Team commit.

Sprint Planning part 2

1. Teams got in place and prepared for the meeting. The meeting should be on a place where Team member can think clear, write down tasks and draw any explanation.
2. Teams should calculate their availability during that Sprint, i.e., Teams estimated the amount of effort they expected for non-product development tasks. These tasks could be, for example, development tasks (laptops, labs,…), organizational meetings, training and knowledge share sessions, reading and helping others or participation in other Scrum meetings, among others.
3. Team picked up USs by priority order and started breaking them down into tasks and estimated each one.
4. Team builds up its own Sprint Backlog with their tasks for the Sprint and post it on the teams' wall.
5. A Daily Scrum meeting was performed in order to synchronize who will start what activity.
6. Finally Team communicated to PAPOs the USs they committed for the Sprint.
7.

Joint Product Backlog Refinement (Joint PBR)

As preparation for this meeting these actions were taken:

- Meeting facilitators reserved the rooms and sent invitations to participants.
- PAPO sent to Teams the new PBL items and PBL items candidate for the next Sprint.
- PAPO and SME prepared at least three Acceptance test examples (i.e., customer requirement specification by examples) per new item.
- PAPO created or updated spread sheet with all items grouped by estimate in order to be used as estimation scale.
- Teams prepared Proof of Concepts (PoC) and Pre-Studies presentations.

The Joint PBR meeting followed then a six step script:

1. On a first step, participants discussed and agreed on possible changes to the Definition of Done (DoD).
2. Next, PAPO presented the spread sheet with the estimation scale.
3. For each new or next Sprint item
    a. Teams showed current status of PoC and Pre-studies, each one with a maximum period of fifteen minutes
    b. PAPO and SME clarified the item in discussion and
    c. Create new Acceptance Test examples, approximately a total of ten examples per item.
    d. The discussed item was estimated during a maximum of five minutes.
4. The fourth step started by splitting items (with more than thirteen SPs) and Acceptance tests for the next Sprint, building up sprint-grained items with approximately eight SPs each. This step was optional if they had less than three items for the next Sprint.
5. Estimation of the split items, with no more than five minutes per item and only one estimator per Team.
6. The last step, was optional, and included a re-estimation of the complete release Backlog, again a maximum of five minutes discussion per item and just one estimator per Team.

Team-level Product Backlog Refinement (Team-Level PBR)

Five step meeting script:

1. Team started by presenting the spread sheet with the estimation scale, i.e., with all stories grouped by estimate.
2. Then, they split items for the next Sprint into fine-grained items of approximately two or three SPs.
3. New acceptance tests examples were created for each fine-grained item, approximately a maximum of twenty examples per item.
4. The new split items were estimated and
5. Created in the PBL where they were linked to other items and features, and old items were marked as duplicated and linked to new ones.

Sprint Review

Sprint Review sequence and steps:

1. Meeting started with a inspect product period of ten minutes:
    a. PAPO shared relevant Business context information during ten minutes maximum. Themes like customer demos, product decisions, competitors and financial information was shared with all Teams representatives.
2. A Sprint inspection step of around two hours started:
    a. PAPO started to explain the Sprint Goals and showed the Definition of Done (DoD) agreed in the Sprint Planning meeting.
    b. For each US item PAPO told if the DoD was met.
    c. If item met DoD is considered Done and PAPO explains its goal.
    d. Team then run automated acceptance tests for that item and PAPO tried it out in the system.
    e. If item was not considered Done, PAPO decided if it can be showed at the end of the Sprint Review.
3. The meeting continues with an Inspect Product for twenty minutes.
    a. New items for the next Sprints were presented by PAPO.
    b. Teams performed comments and suggestions regarding new, changed or deleted items, as well as major impediments that, if removed, would help the Team.

This page intentionally left blank

Annex D – Scrum Practices Summary

Scrum Practice		Frequency/Duration	Participants	Place	Main Challenges/Impediments
Daily Scrum		- Daily in every Team - 15 min. Maximum duration	- Scrum Master - All team members	Teams place	- No impediments reported - Co-located team meeting, no communication and colaboration issues reported
Sprint Planning	**Part 1**	- Sprint 1st Day - 1 hour maximum each part - Part 2 takes place right after Part 1	- PAPOs - 2 representatives of each team	Joint meeting room remotly connected to other sites	- Very interactive meeting difficult to be held remotely. - Communication issues and communication tools limitations reported as main impediments
	Part 2		- All team members - PAPOs should be contactable	Teams place	- No impediments reported - Co-located team meeting, no communication and colaboration issues reported
Joint Product BackLog Refinement		- 2nd Monday of the Sprint - 5 hours maximum duration	- PAPOs - 1 representative of each team - Subject matter experts - 1 faciliator in each site	Joint meeting room remotly connected to other sites	- Reported to be working normally, but participantes felt difficult to held technical discussion remotely and not in their mother language. - Difficulties with different levels on tehcnical knowledge between participants
Team-Level Product BackLog Refinement		- 2nd Tuesday of the Sprint - 3 hours maximum duration	- All team members	Teams place	- Meeting reported to be working very well with all team member around the same table
Sprint Review		- Last Sprint day - 2h30 maximum duration	- PAPO - 1 representative of each team	Joint meeting room remotly connected to other sites	- Reported as a very difficult meetin to be held remotely - Due to reduced common working hours, difficult to fully agree on the meeting schedule - With the increase number of people, meeting duration start to be exceeded. -- Meeting script constantly being updated to optimize meeting
Team-Level Retrospectives		- Last Sprint day, right after Sprint Review meeting - 2 hours maximum duration	- All team members	Teams place	- No much impediment reported due the fact that there were co-located team meetings - Reported issues were the fact that these meeting occurred right after Sprint Review on a Friday afternoon, and some team members desired to leave office early in the afternoon and do not fully commit to the meeting aim. This behaviour was specially seen in Poland site. - Due to time zone differences, India was not able to perform these meeting right after Sprint Review and need to postpone it to Monday.
Joint Retrospectives		- 1st Tuesday of the Sprint, ScrumMasters decide relization if there are new issues to discuss - 1h30 maximum duration	- Scrum Masters - 1 representative of each team	Joint meeting room remotly connected to other sites	- Ideally meeting should occur right after Team-Retrospectives and before Sprint Planning. -- Due to time zone differences it could only be held after Sprint Planning -- There could be taken decisions that will impact the already planned Sprint - Very interactive meeing difficult to held remotely - Verified that they were discussing most the same issues than in the CoPs -- so it was decided to stop these meeting and address the issues to the correspondent CoP
Scrum-of-Scrums		- Every Wednesday intra-component - 30 min. Maximum duration - Every Friday inter-component - 1h maximum duration	- 2 representatives of each team	Joint meeting room remotly connected to other sites	- Meetings realization decided by management, and not from a team's need - Teams didn't reported much interest in these meetings and couldn't reach the meeting aim of sharing information between them to get synchronized and work to get a common goal. - Reported a very weak inter remote team and inter component team cohesion.
Communities-of-Practices (CoP)		- Frequency is decided by each CoP	- CoP members	CoP decision	- Reported as a very as extremely useful when the number of teams in the project start to be high.

This page intentionally left blank

Annex E – Challenges Summary

Hossain Challenges Classification	Case Study Main Challenges	Challenges' Causes	Project Impacts	Strategies to Address Challenges
Asynchronous	- Reduced common working hours	- Time Zone differences between sites - Different working hours culture	- Delays in problem solving (email responses could take 24hours) - Delays in synchronization between teams in different sites - Overall project productivity decrease - Difficult agreements in joint meetings schedule - Impacts in knowledge transfer - Last minute commited code that could break the build and block the still working site	- Meetings schedule synchronization between involved sites - Continuously optimizing joint meetings script, in order to reduce the meetings duration - Concentrate synchronization activities during common working hours
Lack of Group Awareness	- Lack of F2F meetings	- Company travel costs strict restriction	- Remote Communication Resistance - Mistrust Climate between sites - Lack of overall teams cohesion and commitment - Generation of feeling "us and them" between sites	- Use few know-how transfer session and company events to meet face-to-face - Use of Skype Video and Halo Room tools
	- Division by Components (1 Product with 3 different components managed by 3 different POs with 3 ProductBacklogs, developed by 3 different Development units)	- Business/Management decision (out of the scope of this study)	- Created dependencies and interfaces between sites/teams - Created an hidden "waterfall" in the project: -- Serial development -- When a User Story required changes in several components, there is the need to wait for all component to finish their parts to release the User Story - Lack of transparency between components - Difficult to coordinate all component releases - Not possible to load balance work between components - Not very Agile: -- User Stories are seen as E2E customer features (cross component) -- Created dependencies broke Agile principle of open communication and collaboration - Product not seem as one. People focused more on their component work, than in the overall final product goal.	- Intra and Inter components Scrum-of-Scrum meetings
	- Different enterprise cultures	- Involvement of an external/outsourcing company	- Different working mindset: external company though in short term, internal company though in long term - External company focused in showing work done in short term and not in the overall project goal - External company not very Agile and resistant to changes - Inter-team and inter-site mistrust, poor communication and feeling of "us and them"	- Coaching external company for the Agile values and practices
Poor Communication Bandwidth	- Available communication tools limitations	- Internal VoIP tool not good enough for conferences and without video capabilities - Internal VoIP tool not possible to be used to communicate with external company - Poor microphone facilities in joint meetings - Poor video communication facilities	- Difficulties to perform and follow remotely joint meetings - Not possible to emulate efficiently remote persons presence	- Start using Skype for VoIP and Chat communication - Start using Skype video - Halo Room also used later in the project, but not available in all involved sites
Lack of Tool Support	- Continuous Integration inefficiency	- Starting test automation when were already released several product features - Not defined in the beginning clear test development rules and strategy	- Continuous Integration overall system not working efficiently - Huge number of tests developed without sustainability - Delays in code production, because too many broken builds caused by automate tests failures. - Decreased overall product code quality, introduction of errors in the code	- Created a Community-of-Practice to discuss, re-organize and solve issues with Continuous Integration
	- Wiki entropy	- Not defined in project beginning rules and guidelines to post information in Wiki - Not having a dedicated person to maintain Wiki information	- Outdated and not used information kept in Wiki - Same information spread and repeated in different places in Wiki - People never new when other colleague has posted new information - Difficult to find the desired information	- None
Large Number of Project Personnel	- Very fast and early project scaling	- Business decision to shorten product release schedule and try to reach new customers (out of the scope of this study)	- Original project people didn't have their knowledge and experience consolidated enough to coach new people - Continuous integration solution still not working efficiently and product architecture not stable when project scaling started - Initial high decrease in development velocity with slow recovering afterwards	- Ramp-up and knowledge transfer sessions - Experienced team members split and spread over new teams for a smooth new team members integration and know ledge spread.
	- Too many people involved in the project		- High entropy in the project - High need of synchronization between teams - Exponential growth of communication channels - High overhead of non development task, generating low production and de-motivation feeling between the developers	- None
Lack of Collaborative Office Environment	- Initial office organization not very Agile	- Past office organization not planned to be Agile (out of the scope of this study)	- Not much impact, because problems were easily overcome.	- Office space re-organization in order that all teams members were together - Start using flip-chart on the office walls - Creation of permanent joint meeting room
Increased Number of Sites	- Too many sites involved in the project	- Organization decision (out of the scope of this study)	- Communication drawback and differences between countries -- Causing delays in problem solving and synchronization -- Reflected in overall project productivity and time-to-market schedule - Lost Agile principle of open and close communication - Several time zones working in the project. Difficulties in synchronized work. - External companies involved with different enterprise culture - Need of good remote communication tools and remote person emulation tools - Difficult to have very interactive joint remote meetings - Physical distance was an impediment to the team cohesion and team building generating a mistrust climate - High overhead of non development task, generating low production and de-motivation feeling between the developers	- Scrum-of-Scrums Intra-Component remote sites - Scrum-of-Scrums Inter-Components
Other	- Business teams working with a waterfall mindset	- (out of the scope of this study)	- Program and Product Management didn't "talk" the same language as the R&D teams, causing misunderstanding and barriers between them.	- (out of the scope of this study)