Lean Software Development and Agile Methodologies for a small Software development organization

Enrique Moratilla Temprado

Enrique Ruz Bendito
Authors:
ENRIQUE MORATILLA TEMPRADO, enriquemt@gmail.com
ENRIQUE RUZ BENDITO enri.ru@gmail.com

Master thesis
Subject Category:  Quality Technology

University College of Borås
School of Engineering
SE-501 90 BORÅS
Telephone +46 033 435 4640

Examiner:  Roy Andersson
Supervisor,name: Henrik Berlin
Supervisor,address: ABB

Client:  Hannes Göbel, Innovation Lab
Date:  May 27, 2010
Keywords:  quality, IT, organization, Lean, Scrum, XP, agile methodologies
Acknowledgments

When I came to study this Master I was very exciting about the learning opportunity but I saw the end of it far away. However, with hard work and ambition any goal is possible.

There are many people that I would like to remember in this moment, I would like to thank to Hannel Göbel the opportunity to study the Innovation Lab. He was the very first person that talk me about Lean methodologies adapted to software development.

Thanks also to Jan Rohlén because he made the first step, putting me in contact with Henrik Berlin, my thesis supervisor. I will always be grateful to Henrik for the effort and the time spent in meetings and helping us.

To my friend and thessmiate Enrique Ruz, he was the key of the thesis, his help and dedication in the thesis was brilliant. He was the person that put my in contact with Hannes and finally join me in a very difficult moment and thanks him today the thesis is finished.

I would not want to forget all my profesors at the University of Borås, they gave some amazing lessons about quality, environment, risk, change management and above all about the life. Also to my classmates whom helped me to reach here.

Thanks also to all the people that I met since I have been here in Borås, we have enjoyed a lot, many of them have supported me in good and bad moments, thanks to everybody a part of this success is for you too. This chapter of my life is ending in this moment. I have spent a very good time in Borås, now, it is time to move on, I hope do not have to wait to another life to see all you, brothers ;).

Moreover, in this moment I remember many of my friends, I would like to thanks for the supporting when I took the decision of coming here, specially to them that are living abroad or far from their cities. Also thanks the supporting to my very good friends from Indra-Torrejon, my old workplace.

But I am afraid that above everybody my family was the most important during the thesis, I would like to say thank you to Trufa (my dog), to Gema (my sister), to Mariano (my dad), and specially to Flori (my mum), she is the best, without them come to study here would have been impossible. This thesis is also dedicated to my grandparents, Gonzalo and Carmen whom now are having some health problems, I hope the can overcome them.
Abstract

Lean Software Development is a new approach to provide a quality philosophy in software development companies. Lean offers a set of principles that helps companies to identify inefficient processes and waste. Therefore, by applying these principles, we can reduce costs, by shorting development time, incrementing productivity.

In this study a software company with seven employees is analyzed. The company develops software and wants to increase its maturity and the quality of its processes. Currently, it has few documented processes. Furthermore, the company want a common way of working for every project. The challenge relies in adapting the set of principles provided by Lean to this particular case of study.

We seek to analyze the current situation of the company to find out about the problems and limitations of the current way of working. After that we will state recommendations about the use of Lean combined with Agile practices such as Scrum and XP.

As a result we present a proposal for implementation adapted from these philosophies to the needs and characteristics of the company.

We have found that there are several ways in which the company can benefit from the implementation of Lean practices in combination with Scrum and XP. The result is a new framework that can be applied for other small software development companies in a similar situation.

Keywords: quality, IT, organization, Lean, Scrum, XP, agile methodologies, software development.
Contents

1. **Introduction** ......................................................................................................................... 1
   1.1. Aims and objectives ............................................................................................................. 1
   1.2. Scope .................................................................................................................................. 1
   1.3. Method used ........................................................................................................................ 1

2. **The company – Innovation Lab** ............................................................................................... 3
   2.1. Description ............................................................................................................................ 3
   Main activities ............................................................................................................................ 3
   Core values ................................................................................................................................. 3
   2.2. Organizational chart ............................................................................................................. 4
   Staff ........................................................................................................................................... 4
   Management team ...................................................................................................................... 4
   2.3. Current projects .................................................................................................................... 5
   TFU ............................................................................................................................................ 5
   VFU (Transit) ............................................................................................................................... 6
   QlikView .................................................................................................................................... 6
   e-Me ........................................................................................................................................... 6
   PsYoungSupport ......................................................................................................................... 6
   2.4. Other projects ....................................................................................................................... 7
   2.5. Planning and Management tools .......................................................................................... 8
   2.6. First impressions .................................................................................................................. 10

3. **Methodologies** ........................................................................................................................ 12
   3.1. Introduction .......................................................................................................................... 12
   3.2. Lean Management Software ............................................................................................... 12
   Why Lean Software Development? - Lean as philosophy ......................................................... 12
   The Seven Lean principles ......................................................................................................... 12
   Principle 1 - Eliminate waste ...................................................................................................... 13
   Principle 2 – Amplify learning .................................................................................................... 14
   Principle 3 - Decide as late as possible ....................................................................................... 16
   Principle 4 - Deliver as fast as possible .................................................................................... 17
   Principle 5 - Empower the team ............................................................................................... 18
   Principle 6 - Build Quality in .................................................................................................... 20
   Principle 7 – Optimize the Whole ............................................................................................. 20
   3.3. Scrum ................................................................................................................................... 21
   Why Scrum? ............................................................................................................................... 21
   Benefits of Scrum ....................................................................................................................... 21
   Purpose ..................................................................................................................................... 22
   Scrum Theory ............................................................................................................................ 22
   Scrum Framework ..................................................................................................................... 22
   Scrum Roles ............................................................................................................................... 23
   Time-Boxes ................................................................................................................................ 24
   Scrum Artifacts ........................................................................................................................... 25
   Sprint Backlog ............................................................................................................................ 25
   3.4. XP ....................................................................................................................................... 26
   Why XP? ................................................................................................................................... 26
   What is XP? ............................................................................................................................... 26
   The XP Process .......................................................................................................................... 27
   Test Driven Development TDD ................................................................................................. 29
1. Introduction

Quality systems usually provide a good framework for good processes, but they do not describe the particular deployment for every process. This work is up to the company, for the large companies the implementation requires a big effort in terms of documentation and bureaucracy, but the effort is even bigger in a small company, due the limitation of its resources.

1.1. Aims and objectives

According to Wayne Goddard, et al. (1996) creative research comprises of both practical and theoretical researches. The main objective of this report is practical, to adapt the Lean principles combined with the practices of Scrum and Extreme Programming (XP) to a small organization working within the Business and Informatics Department at the University of Borås called InnovationLab. By performing such study, a theoretical approach is also created and can be, eventually applied to different organizations that share the same principles of InnovationLab, SME companies developing innovative solutions. The framework proposed in this study can also be set as a future goal for an analysis of implementation in bigger organizations.

1.2. Scope

We intend to use the set of Lean Software Development principles adapted by the Poppendiecks from Lean Manufacturing (Poppendieck) (Poppendieck, 2003), focusing our efforts in finding and eliminating waste sources. We will provide a framework combining Lean, Scrum and Extreme Programming techniques that will set the main structure for managing company. This framework will help preventing waste generation.

We will focus in how to apply these techniques to the InnovationLab in particular. Although this framework could be used for organizations similar to InnovationLab, an analysis of the company should be performed prior to the implementation of such a framework.

1.3. Method used

The first stage followed in this research is the learning process through the read of literature about Lean Software Development and about Agile Methodologies, in particular Scrum and Extreme Programming.

The second stage is the data acquisition that is depicted into three different phases. The analysis of the information gathered will start with the first phase, and will continue throughout the entire data acquisition process to create a more accurate data gathering.

The data acquisition will be performed in three different phases:

- 1st Phase: We will interview the manager in order to get the first impressions and the general picture of the company, as well as the general management methods. After that, two interviews with different developers will be performed in order to know the development process and the team behavior.
- **2nd Phase**: After the first interviews and a better comprehension of the Lean principles, we will conduct a survey among all developers at the InnovationLab. The goal of this survey is to gain knowledge about the overall way in which the team works.
- **3rd Phase**: After the analysis of the survey results, we will have a final interview with the manager of InnovationLab in order to improve the accuracy of our analysis.

The analysis consist of realize about the findings from the data acquisition and to match them with the Lean principles in order to provide some tips and suggestions for the implementation of a methodology of work that will improve the efficiency and will ease the management effort of the organization.

The last stage will consist of the creation of a proposal for implementation that will provide ideas on which is the best way to implement the framework proposed.
2. The company – Innovation Lab

2.1. Description

“Without passion, no innovation”

Innovation Lab is an organization working within the University of Boras that provides researching tools and completely functional solutions to researchers, research groups and other organizations (InnovationLab).

Main activities

Research Support
Innovation Lab provides support for researchers developing tools that they can use in their investigations. Due to the fact that InnovationLab is part of the University it provides better understanding of the researcher's needs.

Research ideas can usually be beyond only theory. Sometimes there is a value and a strong need to realize an idea for a pilot, prototype or a finished product. They try to depict the research support in three stages:

- **Planning**: studying the customer, their necessities.
- **Implementation**, executing the idea, model, specification, algorithm.
- **Production**, releasing the project, generating documentation and knowledge.

Operations Support
InnovationLab provides tools to state, regional and municipals departments, developing system support operations:

- **Planning** as business / concept analysis, decision support, claims status, project.
- **Procurement Support** as an expert assessment of ERP / standard systems, decision support.
- **Implementation** creating new ideas, models, algorithms.
- **Production** releasing tools those are useful for the community.

Core values

The company is focused in innovation, especially towards innovation in IT. They explain innovation as:

- The use and further development of new concepts, methods, techniques and tools; or the positive change the end user notices. The change that ensures a better, easier or more effective use of the artifact in question.
2.2. Organizational chart

*Staff*

The body of the InnovationLab is built in 7 people. It is a very flat organization that tries to erase the role’s labels. There is only one person who stands out from the others, the manager. The manager is currently working 70% of his time in two projects; the rest of his time is spent in: Customer acquisition, financial management (where he receives some help from another person in the University), human resources and general management.

The manager delegates the project management for many projects, assigning to the developers the projects and allowing them to be free enough to work in their way. It means that most of the developers have also to deal with the management of one project.

![Organization Chart](image)

*Figure 1 Organization Chart*

*Management team*

InnovationLab have a management team composed of five people who in some way are related to the business. Management team's task is to act as an advisory body for the activity and its director. The management team for InnovationLab included the following people:
This management team meets four times per year with the manager of the InnovationLab. But they do not work 100% on the company; they work as an advisor/consultant board. Also they try to find new customers.

2.3. Current projects

The aim of this section is to show the reader the different projects that the company has nowadays active. The section has a deeper description of the larger projects because they will require more time and resources.

The end of the section contains projects that because its size or its status are considered less important.

**TFU**

TFU (Tjänstefördelning och Uppföljning) is a product to planning teachers’ schedules and work hours. The main customer is the University of Borås, but it has actually been offered to several universities in Sweden. The users of this program are administration staff and university lecturers; right now the number is around 200.

It is a web-based system, which is developed using C# and is deployed in the IT-system services department. It uses Extreme Programming as a methodology. Three developers are currently involved in this project, but they are not maintaining the hosting.

The project was released and now they are working in university requests there is no real knowledge about the deadline.
VFU (Transit)
VFU is the acronym for “Verksamhetsförlagd utbildning”. It is a tool for managing students’ internships in medical centers. The main customer is the University of Borås.

They developed the project using ASP.Net with AJAX. The project finished in 2008 and they are currently maintaining it. There are two developers in charge of this project. The project is based in the same system of TFU, and then the number of users and the characteristics about how to deploy it are the same. The first release was two years ago.

QlikView
QlikView is a professional Business Intelligence tool for financial management for the University of Borås. This tool needs to be customized and set ready for the financial department at the University. Only one developer is working on this project, small time is required.

This customization is done with a proprietary scripting language of QlikView. InnovationLab does not own the code of this program; they only make scripts with new functionalities which are requested by the University of Borås.

e-Me
e-Me is a research project that originally aims to create a personal electronic assistant that university, businesses and governments need to contact if they want you anything. e-Me can take care of all your e-mail accounts, manage administration, manage calendars, communicate with your mobile phone, looking for interesting offers, and more.

The goal or scope of this project has significantly changed over time due to the change of the requirements. It is currently aimed for different users and not only students.

Nowadays there are between three and five developers working on this project. This is the most important project of the InnovationLab.

The project was started in 2006 and there is not a fixed deadline. This project is not released yet and the potential number of user is really big, the idea is that can be used for everyone.

PsYoungSupport
The overall objective of the research project PS Young Support is to evaluate ICT interventions (information and communication technologies-logical) in the form of information and support to young people who have a relative with poor psychological health.

The working period is estimated from 2008-12-01 till 2010-05-31.

The technology used in this project is a CMS (Content Management System) called Drupal. The main customer is the University of Göteborg. The number of developers involved in this project is only one.
2.4. Other projects

This is a list of other current projects InnovationLab takes care of, but are not as big or important as the others:

- **Nywebb**
  - The portal of the University of Borås ([www.hb.se](http://www.hb.se))

- **Boråsstad**
  - A web application to be able to hire apartments in Borås.

- **Kursinfo**
  - The application used at the university to keep track of all courses and programs.

- **Effit**
  - Building a community for

- **Bättre konceptet**
  - Web application that keeps track of Education for the employees of all types of companies

- **Intime**
  - The time reporting system at InnovationLab
### 2.5. Planning and Management tools

The current planning is showed above. As you can see it is divided in 7 parts which correspond with the different developers and the manager of the InnovationLab. It seems that there is a tendency in assigning many projects during the same period of time. Even if the time dedicated to these projects is small the time switching between each other could be important for the developer. For example, developers 3 and 4 have at least 3 projects during the same period of time. It also increases the chances of not having the developers alienated during development iterations, this means, not having them working in the same project.
As the pareto-chart is showing us, the larger projects in InnovationLab are E-Me and TFU. Working on them, around 45% of the resources of the company are spent. Another important issue are the PhD Students/Developers, they are not working 100% of their time in the company. As the chart shows, 17% is dedicated to research ("forskning") tasks. In management tasks the company spends around 5% of their resources. Right now the amount of time is enough, but if the company wants to grow it should increase the time dedicated in management.
2.6.   First impressions

In this section we will have a brief discussion about the peculiarities of the information shown above. The reader could be interested in take a look to the appendix, where you can find the questionnaires and the answers.

After a thorough information gathering of the current organizational structure and composition combined with a review of the current projects Innovation Lab is working on, we have identify the necessity of a deep definition of their methodology of working in order to manage the multiple projects.

These projects are difficult to manage due to their special characteristics. Most of them have a very short development time, with very different goals or fields of study; and development technologies. For example, in the TFU project, some of the issues that we have seen during the interviews are the constant change in the requirements and the requirement of the University of Borås to make frequent releases as soon as the code is ready.

Furthermore, the staff is composed by six developers and one manager. This manager sometimes helps in development issues as well. The roles for the employees are not fully fixed, i.e. during lifetime development the Scrum-Master role has change in a couple of situations.

It seems as if there is not a clear set of rules to drive the path that the organization has to follow and motivate the staff.

Some problems arise when we have to deal with projects with these characteristics:

- Due to the limited resources of the company, the dealing with many different projects, even if they are short-time projects, can create a lot of time-spending when it comes to track the projects. There is a need of assuring an efficient project tracking method.

- There is also a concern when it comes to a single resource (developer) managing many different projects of very different scope and technologies. The effort that a developer must put in handling different tasks affects the overall performance of his /her job, in our opinion.

- Due to the amount of different tasks each developer has to manage, we believe there is a loose of scope of the projects. That means, having to work in different tasks inside different projects, makes the overall view of a project fuzzy. That, in turn, can lead to forgetting the requirements or needs of the customer or even the goal of the project.

- From our first meetings with the manager, we realize that one of the problems for him was the lack of a standard documentation. There is not a standardized way to write the reports in the company. The manager spend a lot of time adapting the documents and reports to a common structure and appearance.
Another requirement from the manager was the need to implement Test Driven Development methodology in all the projects. Right now it is implemented in some projects with different acceptances from the employees.

In some projects, the customer is not clearly identified, for example, in e-Me the developers of the project have to make up the requirements, there is not project owner. This situation can lead to a constant change of the requirements.
3. Methodologies

3.1. Introduction

This chapter idea is to be a guide of the current methodologies that the researcher is going to use during the Analysis and Proposal phases. After comparing many methods we focus on this three: Lean, Scrum and XP. The reader can find further information about the reasons that lead to choose them at the beginning of each section during this Chapter.

3.2. Lean Management Software

Why Lean Software Development? - Lean as philosophy

When we were trying to find a good quality method to apply to a company Lean Software Development stand out for many reasons, but the main reason was the cost of implementation. In the beginning we were considering ISO 9000 and CMMI methods, but those methods attempt to remove process design and decision-making authority from developers and put them under control.

The main idea of Lean is not to add more activities to the company (such as project tracking and control), but is basically waste elimination and the belief that you can improve your production using less resources. In this case study, the company to improve is very small (read more information in Chapter 1) and the resources are limited.

Today, lean manufacturing sets the standard for discipline, efficiency, and effectiveness. In fact, using lean principles creates a competitive advantage that can be really difficult to copy. It means that Lean is much more than a philosophy; it is a way of thinking and because of this requires a big effort, at the beginning.

Lean Production is actually the new name given to the Toyota Production System (P.Womack, o.a., 1990), some companies have tried to adopt this approach but it is difficult. Like in every change, resistance from those invested in the old model is fierce (Bovee, 1998).

The Seven Lean principles

“Principles are underlying truths that do not change over time or space, while practices are the application of principles to a particular situation. Practices can and should differ as you move from one environment to the next, and they also change as a situation evolves.” (Poppendieck, 2006)

These seven principles of Lean Software Development are based in the Lean manufacturing methodology, but adapted to the idiosyncrasy of a software developing environment.

---

1 This document section is based on the books (Poppendieck, 2006) (Poppendieck, 2003)
**Principle 1 - Eliminate waste**

This is the first step to develop lean thinking. To eliminate waste, you first have to recognize it. Waste is anything that does not add value, then, the aim should be focus in develop a sense of what value really is.

The most important resources of waste come from:

- **Partially done work**
  - Uncoded documentation.
  - Unsynchronized code.
  - Untested code.
  - Undocumented code.
  - Undeployed code.

- **Extra processes**
  - Create unnecessary paperwork. If you need to produce something that adds little customer value, there are two rules to remember: Keep it short, keep it high level.
  - To evaluate whether a process is interesting just check if someone is waiting for it.

- **Extra features**
  - Things added into a system just in case that they are needed.
  - High possibility to become obsolete.

- **Task switching**
  - Waste in switching, work moves faster though a pipeline that is not filled to its capacity.

- **Delays**
  - In waiting for project approvals, people assignation, people availability, debug to pass tests.
  - Keeps the customer from receiving value as quickly as possible.

- **Motion - Handoffs**
  - Find the resources (even other person of the staff) quickly.
  - Documents that go from one person to another and do not contain the information that the other person needs. A move artifact between groups is a big source of waste.
• Defects
  
  o To reduce impact, test immediately, integrate often and release to production as soon as possible.

*Principle 2 – Amplify learning*

Lean principles cannot be applied directly because generation of good software is not a production process; it is a development process.

In development, iteration generates value because it is similar to design a recipe, in fact, is a learning process involving trying and error processes. Quality design means realization of purpose or fitness of use rather than performance requirements.

Development is not intended to produce repeatable results; development produces appropriate solutions to unique customer problems.

There are two schools in developing software:

1. Focus in being sure that each design and each segment is perfect from the 1st time.

2. Having a small, rapid, try-it, test-it, fix-it, cycles to make sure the design and code are ok.

*Learning cycles*

Usually the problem to be solved is understood best by the people in the business with the problem. Working screens generate better knowledge and faster compared to a requirement document.

Iterations and refactoring are effective to generate knowledge. Then, it is good to have many short learning cycles. It seems the best structure for ill-structured problems (like E-me).

*Feedback*

• For unforeseeable events, feedback is needed.

• Increase feedback is the most effective way to deal with troubled software development projects and environments.

• Some good practices could be:
  
  o Run test as soon as code is written, do not let defects accumulate.
  
  o Check out ideas by writing code. No adding more documentation.
  
  o Show to customer an assortment of potential user screens and get their input. Instead of ask him and get more requirements.
  
  o Instead of study carefully which tool to use, bring the top 3 and test them.

• Whenever paperwork should be done, it must be for an immediate customer, someone eager to make use of the results of the work.
Iterations
Iteration is a useful increment of software that is designed, programmed, tested, integrated and delivered during a short fixed timeframe.

Iteration planning
- Planning session occurs to estimate the level of difficulty.
- Should have a fixed time-box, time duration must be different in each case of project.
- Development team must be free to accept only the amount of work that they think can do.

Team commitment
- Should be small and staffed with the necessary expertise.
- Must have enough information and resources.
- Freedom to do its work.
- Must have a basic environment for good programming
  - Automatic build process
  - Automated testing
  - Coding standards
  - Version Control Tool.

Negotiable scope
- 45% of features in a typical system are never used.
- 19% of features in a typical system are rarely used.
- Minimizing requirements, highest priority features.
- Acceptance test, written and passed.
- Use of burn-down charts could be included

Synchronization
Common code ownership requires that several people be synchronized.
- Build system every day, after work done by developers.
- Check-in new code, build and run automated test (“daily build and smoke-test”)
- More builds, more feedback is better (builds should be automated)
- If the takes too long run during the night on in weekends.

Keep it simple by doing it as often as possible; the goal is to have workable code at the end of every day.
Set based Development

- Communication is about constraints, no-choices, no solutions.
- The method is to develop multiple options, communicate constraints and let solutions emerge.
- It adds new dimension, during early iterations, multiple choices are developed for key features; in later iterations they are merged or narrowed to a single choice.

Principle 3 - Decide as late as possible

Once early decisions are made, other decisions are built on terms making difficult to change. Premature design commitment is a design failure mode that restricts learning, exacerbates the impacts of defects, limits the usefulness of the product and increases the cost of change.

It takes expertise to know which options to keep open. Options do not guarantee success; they set the stage for success if the uncertain future moves in a favorable direction. Allow fact-based decisions on Learning rather than speculations.

Concurrent software development starts when only partial requirements are known. Using developing in short iterations, then provide the feedback that cause the system to emerge.

Applying this method is possible to delay commitment until the last responsible moment, the moment at which failing to make a decision eliminates an important alternative.

Tactics

- Share partially complete design information
  - Good design is a discovery process, done though short, repeated exploratory cycles.
- Organize for direct, worker-to-worker collaboration: release the information between people that understand the details of what the system must do to provide value to the people to understand how the code works.
- Develop a sense of how to absorb changes; repair errors before they cause problems.

Making decisions – Depth First vs. Breadth First

Depth-First is based in early commitments, if a change is necessary is bad;

- Good in stable domains with, expert teams.
- Breadth-First is based in delaying commitments;
- Does not need a stable domain.
- Because is domain is expected to evolve.
- However, is effective even when domain is stable.
- Its better when exist a lack of knowledge or assurance or when taking decision could bring the project to obsolete solutions.

Intuitive
In the real world there are many examples of team that develop its job with success based in an intuitive way of thinking. It is easy to find military teams that are focus in the intuition. We can stress some of them ways of thinking.

- They plan but not predict.
- Understand the essence of the situation and evaluate its strengths and weaknesses.
- On-ground soldiers make decisions, not their officers.

**Simple rules**

- Allow everyone in an organization to act quickly, synchronously in a coordinated manner, without the necessity of waiting for instructions from above.
- Allow decisions to be made on spot, when and where they are needed.
- In SW development give people a framework to making decisions.
- Should be few key principles (like the Seven Lean Principles).

**Principle 4 - Deliver as fast as possible**

*“Haste Makes Waste”*

Deliver as fast as possible is an operational practice that provides a strong competitive advantage. It complements “decide as late as possible”. The faster you can deliver, the longer you can delay decisions.

**Pull systems**

People should do to make the most effective contribution to the business. There are two ways to make workers more effective:

- Tell them what to do.
- Set things up so they can figure it out for themselves (fast-moving environment).

**Software pull systems**

- Make the work self-directing.
- The system is similar to Scrum. Cards are not assigned to developers, they choose the cards.
- It is important to have a regular brief meeting, preferably daily:
  - It should not last more than 15 minutes.
  - Everyone in the team should show up.
  - Tasks:
    - Summary of what they did yesterday, and the plan for today.
    - Where they need help.
    - Leader must run interference for the team (any support issue).
- Requires short time-boxes (i.e. 1 month) or it can degenerate to a push system.
- One of the features is visual control or management by sight.
Cost of delay

“A dollar saved is a dollar earned”

For example if you have to decide whether or not to apply a tool make a model of product and economical impact, not only in the current period, also in a long term.

Every week or day in delay can cause a big impact in costumer company economy.

It is useful to create a place where everyone can come and see:

- What has already been done?
- What is being done?
- What has not been done?

Ask the team to focus on the way they spend the time.

Find the longest queues in the area and chart the cycle of each job.

Pick the biggest bottleneck and form a bottleneck force

- Find the way to reduce it.

Principle 5 - Empower the team

An organization that respects SW developers as professionals will expect them to design their own jobs with proper training, coaching and assistance. They will improve as a part of a learning process.

Self-Determination & Motivation

Assume that the employees know how to do their work and trust them and let them do their jobs.

Self-organizing group that become passionate about a possibility give better results:

- “Hire good people, and leave them alone”
- “If you put fences around people, you get sheep”
- “Encourage, don’t nitpick. Let people run with an idea”
- “Give it a try – quick!”

Purpose makes work energizing and engaging:

- “People suffer when they lack purpose”
- Intrinsic motivation is very powerful if a team commit together. Some rules:
  - Start with a clear and compelling purpose.
  - Be sure the purpose is achievable.
- Give the team access to customers.
- Let the team make its own commitments (iterations)
- Management role is to run interference. Help in the team demands.
- Keep skeptics away.

The Building blocks of motivation

- **Belonging:** being honest, respecting the staff, team must win or lose as a group.
- **Safety:** let people work, even if the commit fails, it is better in the long-term.
- **Competence:** SW development environment must be disciplined. Basic practices (like controlling code repository, coding standards, build processes, automated tests) are required for rapid development. Also, sharing ideas and improving designs.
- A sense of competence comes from knowledge and skill, positive feedback, high standards and meeting a difficult challenge. A leader who delegates and trust on the team but also verify that workers are on the right track and guide them.
- **Progress:** Members need to feel they have accomplished something. It is another reason to make iteration (compelling itself). Members must celebrate objectives in relation with its importance.

Balance between personal and professional life increase performance. Furthermore when an emergency comes up the members can rise up the dedication and solve the problem.
Principle 6 - Build Quality in
According to Shigeo Shingo, there are two kinds of inspection: inspection after defects occur and inspection to prevent defects (Shingo, 1989). The idea is to fix the errors immediately after they are found, finding its cause. In other systems the errors are tracking and corrected afterwards. That is a waste of time.

The job of tests is to prevent defects, not to find them. A quality assurance manager should aim in build quality from the start in the code. Usually most of the mistakes are caused because system allowed the mistakes to happen. Therefore, majority of defects are actually management problems. The slogan, “Do it right the first time” in software development is used to be sure the code behaves exactly as intended using test-driven development and continuous integration.

Principle 7 – Optimize the Whole
A lean organization has to optimize the whole value stream, from the time it receives an order to address a customer need until software is deployed and the need is fulfilled. It is necessary to consider the whole stream, otherwise the performance will suffer.

In software development is very common to find a tendency to suboptimize:

- New features problems:
  1. A customer wants some new features, as soon as possible.
  2. Developers have to work on it fast at any cost.
  3. Result
     - Changes are made in code base careless.
     - Complexity of the code is increased.
     - Rise of number of defects.
     - Exponential increase in the amount of time to add features.

- Testing problems:
  - Testing is overloaded with work.
  - Result:
    - Testing occurs after coding.
    - Developers do not have immediate feedback.
    - Developers create more errors.
    - Testing time is increased.
    - Repeat the cycle.

As a remedy we recommend use Test Driven Development, about we will talk in the next sections.
3.3. Scrum

Why Scrum?
One of the first reasons is that the Innovation Lab and its staff have already the knowledge to work with Scrum since it is the methodology for some of their current projects.
It is much easier, faster and effective to implement a working methodology whether the people who are going to work with it know the principles or, in this case, has deeper understanding on the principles of this method. Furthermore, we want Scrum to be the main methodology that leads all the projects with a main but significantly changed product backlog. We will explain this idea later.

Another obvious reason is because of the broad acceptance of the methodology and its implementation in many other organizations. Successfully companies like Microsoft, Oracle, Palm, Sun or IBM are currently using Scrum to manage their software development process.

There is a need to keep a correct track of the many different projects Innovation Lab works in without making this tracking effort very time consuming. The several tools Scrum serves the organization, can help them to prioritize the task they must work on, keeping in mind the most important tasks but without forgetting about the rest and without involving a lot of time consuming. Scrum is also a tool that helps the employees to acquire a regularized rhythm or work. This is very important in Innovation Lab due to the different size of the projects they work on. A stabilized and systematic deadlines for delivery, helps the development team to know better how to distribute their time in order to reach the deadline with a finished code.

Benefits of Scrum

Scrum can benefit InnovationLab in several ways, although Scrum tends to increase more the efficiency when managing bigger and cross-functional teams, Scrum can still contribute the overall efficiency and has one main advantage, bring a consistent structure to manage all the projects.

"Industry data shows that about half of the software features developed are never used, development can be completed in half the time by avoiding waste, or unnecessary work.” (ScrumAlliance, 2010) we have Scrum running after a few iteration, the team will start delivering higher business values first and will forget about features that are likely to not been used, if in the end they are needed, they will be asked by the customer.

Scrum decreases the management efforts in the teams. By allowing the developers to choose the tasks they want to commit to. This activity brings motivation and commitment to the team. With the help of the daily Scrum meetings, the impediments are identified in early stages, allowing the team to face and solve them before they become more complex. The impediments will be prioritized and removed, increasing the productivity.

By using the rules that Scrum gives we are encouraged to perform continuous inspections, reorganization, adaptation and self-organization. We allow the customer to actively participate in the product they will eventually use. This brings two clear benefits, a most accurate definition of what they want, preventing the development of functionalities not required or different; and gives the customer a higher quality service or product in the first release.

---

2 This section is based in (ScrumAlliance, 2010)
The visibility of the projects is increased since the structure is available for those in and outside the projects. The idea behind Scrum is that the requirements often change, when there are long term planning, is more difficult to face to changes in the requirements, therefore, Scrum seeks to solve the problem of changed requirements.

**Purpose**

Scrum is not a process or a technique for developing software; rather, it is a framework within which you can employ various processes and techniques. The role of Scrum is to surface the relative efficacy of your development practices so that you can improve upon them while providing a framework within which complex products can be developed.

The idea behind Scrum is that the requirements often change, when there are long term planning, is more difficult to face to changes in the requirements, therefore, Scrum seeks to solve the problem of changed requirements.

**Scrum Theory**

Scrum is an agile methodology, a process for incrementally building software in complex environments. Three pillars uphold every implementation of empirical process control:

- **Transparency**: Scrum ensures that the development process must be visible to those managing the outcomes.

- **Inspection**: The different aspects of the process must be inspected frequently enough so that even slight variances in the process can be detected.

- **Adaptation**: If the inspector determines that one or more aspects of the process are outside acceptable limits, the inspector must adjust the process or the material being processed. The adjustment must be made as quickly as possible to minimize further deviation.

There are three tokens for inspection and adaptation in Scrum. The Daily Scrum meeting is used to inspect the progress toward the Sprint goal, and to make changes that optimize the value of the next day. In addition, the Sprint Review and Planning meetings are used to follow up toward the Release Goal and to make adaptations that improve the value of the next Sprint.

**Scrum Framework**

The Scrum framework consists of a set of Scrum Teams and their associated roles; Time-Boxes, Artifacts, and Rules.

A **product owner** compiles all the requisites for the product and prioritizes the tasks involving the development of the product.

The result of the Product Owner’s work is posted in the **Product Backlog** which is a to-do list that can be constantly reprioritized. Before the beginning of a spring, the highest prioritized tasks are transferred to the **Sprint Backlog**.

Together with a customer, the project members form a **Scrum Team**. During the discussions with the Product Owner, the goal of the sprint is determined and the tasks are distributed.
The **Scrum Master** trains the development team; he removes any impediments and works constantly to ensure that the team can work with the highest efficiency possible to reach the goals stated in the Sprint.

By the end of each Sprint, the product’s market value must grow and new functions and improvements can be delivered to the customers.

**Scrum Roles**

**The Scrum Master**
The Scrum-Master is responsible for:

- Ensuring that the Scrum Team adheres to Scrum values, practices, and rules.
- Helps the Scrum Team and the organization adopt Scrum.
- Teaches the Scrum Team by coaching and by leading it to be more productive and produce higher quality products.
- Meets with the team every day in brief meetings called **Daily Scrum**.
- After each Sprint, the Scrum Master holds an **Evaluation Meeting** with the team during which experiences and conclusions are reviewed.

**The Product Owner**
The Product Owner is the only person responsible for managing the Product Backlog, ensures the value of the work the Team performs and ensures that it is visible to everyone. Everyone knows what items have the highest priority, so everyone knows what will be worked on.

Represents the customer’s voice and makes sure the team will work with the correct business perspective.

The Product Owner is often a customer, but can also be an employee in the organization. This person must have comprehensive knowledge about engineering, marketing and business processes.

**The Scrum Team**
They perform the actual work. Teams are cross-functional; Team members must have all of the skills necessary to create an increment of work. Team members often have specialized skills, such as programming, quality control, business analysis, architecture, user interface design, or data base design. However, they should be able to swap task. There are no titles on Teams, and there are no exceptions to this rule. Teams do not contain sub-Teams dedicated to particular domains like testing or business analysis, either.

The optimal size for a Team is seven people, plus or minus two. When there are fewer than five Team members, there is less interaction and as a result less productivity gain. If there are more than nine, too much coordination is required. Team composition may change at the end of a Sprint.
**Time-Boxes**

**The Sprint**
A Sprint is an iteration of between 2 to 4 weeks. Sprints are time-boxed. During the Sprint, the Scrum Master ensures that no changes are made that would affect the Sprint Goal. Sprints contain and consist of the Sprint Planning meeting, the development work, the Sprint Review, and the Sprint Retrospective. Sprints occur one after another, they never overlap and they are chained.

Scrum is a framework for a project whose horizon is no more than one month long, longer horizons are risky. During a Sprint, the Scrum Team is self organized and works under its own responsibility.

**Planning Meeting**
The Sprint Planning meeting is when the iteration is planned. It happens the first day of the one month Sprint. During this meeting, the Scrum Team must decide how to work with the tasks given by the Product Owner.

Having selected the Product Backlog, a Sprint Goal is set. The Sprint Goal is a goal that will be met through the implementation of the tasks in the Product Backlog.

**Sprint Review**
It is held at the end of the Sprint. This is a four hour meeting for one month Sprints. During the Sprint Review, the Scrum Team and stakeholders collaborate about what was just been done. They talk about what are the next things that could be done. This is an informal meeting, with the presentation of the functionality intended to foster collaboration about what to do next.

**Daily Scrum**
Each Team meets daily for a 15-minute inspect and adapt meeting called the Daily Scrum. The Daily Scrum is at the same time and same place throughout the Sprints. During the meeting, each Team member explains:

1. What he or she has accomplished since the last meeting;
2. What he or she is going to do before the next meeting; and
3. What obstacles are in his or her way.

Daily Scrums improve communications, eliminate other meetings, identify and remove impediments to development, highlight and promote quick decision-making, and improve everyone's level of project knowledge.

The Scrum Master makes sure that the Team has the meeting. He or she teaches the Team to keep the Daily Scrum. He must help the Team members to solve the problems they have found during the previous day.
Scrum Artifacts

Product Backlog
It is a list of the requirements of the product that the Team(s) is developing. The Product Owner is responsible for the Product Backlog, its contents, its availability, and its prioritization. These requirements can be new functionalities or bugs. The Product Backlog is never complete, it evolves as the product and the environment in which it will be used evolves.

The Product Backlog represents everything that is necessary to develop and to launch a successful product. It is sorted in order of priority. The higher the priority is, the more urgent it is, the more it has been thought about, and the more consensus there is regarding its value. Tasks with higher priority are clearer and have more detailed information than lower priority tasks.

Sprint Backlog
Consists of the tasks the Team performs during a Sprint. Many are developed during the Sprint Planning Meeting. It is all of the work that the Team identifies as necessary to meet the Sprint goal. Sprint Backlog items must be decomposed enough so changes in progress can be understood in the Daily Scrum.

Demonstration and Evaluation
Each Sprint finishes with a demonstration of the running software to all the stakeholders. This is the Evaluation meeting that in turn is the starting point for the next Sprint.
3.4. XP

Why XP?

XP can be described as a set of rules rather than a methodology of working. It deals more with the way the developer or the team of developers proceed while performing their duties. Due to this difference of concept, XP can be seen as a way to add value to a methodology. We decided to implement XP tied to Scrum for the Innovation Lab due to the nature of its rules. Rules that allow the developers to improve the efficiency of their work by performing more collaborative development.

Innovation Lab is, unlike many other organizations, an organization that acquires small new projects, sometimes projects that would not be profitable for some other organizations. This projects, due to their different nature, can be very different and, therefore, need new knowledge in order to develop it. The process of acquiring new knowledge can be done by either running a lot of prototyping or studying the possible solutions. Thanks to Lean Software Development, we now know that the best practice to follow is by prototyping, because that code used in prototyping will be partially used in the future implementation. XP and prototyping can be a good thing to merge because prototyping is also a brainstorming process, and this process requires more than one person to make it work.

Another advantage of the XP tools is the need to integrate and test very often. This is of vital importance because of the need of the developers to change from project to project quite often. Frequently integration makes the code more stable and makes it easier to get back on it after coming from a different task.

What is XP?

Extreme Programming is a set of rules and processes to develop software; it is frame under the Agile methodologies. XP is intended to improve software quality and responsiveness to the change of customer’s requirements. Since is an Agile methodology, it tries to release business value in short development cycles, which is intended to improve productivity and introduce checkpoints where new customer requirements can be analyzed.

---

3 Based on (Jeffries, o.a., 2000)
The XP Process

Exploration
During this stage we document the customer’s requirements. These requirements must be sorted by their business value by the customer; there is no need to explain deeply the requirement, before each iteration the customer will depict the story and will provide an acceptance test. Each of these requirements must be written in a Story Card with a short explanation for the test.

Spike
A spike solution is a very simple program to explore potential solutions. The idea behind it is that instead of spending time analyzing or planning, teams should try to create or fast experiments. Spikes are often used when there is a need to estimate the time needed for something the team members have never done before. Instead of guessing and maybe doing a bad estimation, they should conduct an experiment. Even if they spend too much time doing the experiment it will be useful for the later implementation.

Release Planning
A release planning is a meeting used to create a release plan, a design of the overall project. The release plan is then used to create iteration plans for each individual iteration. These releases usually take between one to three months.

The idea of the release planning meeting is for the development team to estimate each user story in terms of ideal programming weeks. An ideal week is how long you implementing a

---

4 The customer creates scenarios to test when a user story is correctly implemented. A story can have one or many acceptance tests. Customers are responsible for verifying if the acceptance tests are correct and review test scores to decide which failed tests are of highest priority.
story are going to take if the team has nothing else to do. The customer, then, decides what the most important story is or which has the highest priority to be completed.

User stories are printed or written on cards. Together developers and customers choose a set of stories to be implemented in the following release. The estimation of time for each story is written in the card.

You may plan by time, determining how many stories can be implemented before a given time, or by scope, how long a set of stories will take to finish. Individual iterations are planned in detail just before each iteration begins and not in advance.

Team cannot commit to do more job than they did in the previous release.

**Iteration Planning**

An iteration planning meeting is called at the beginning of each iteration to prepare the tasks. Each one lasts between 1 to 3 weeks. User stories are chosen of the most valuable to the customer first. Failed acceptance tests to be fixed are also selected. The iteration length is chosen and never changes.

Based on the estimations of the team and the amount of work they can commit to in iteration, the customer chooses the stories that will be completely implemented.

The user stories and failed tests are broken down into the programming tasks that will support them. Tasks are written down on index cards like user stories. While user stories are in the customer's language, tasks are in the developer's language. Duplicate tasks can be removed. These task cards will be the detailed plan for the iteration.

**Coding**

Software is written by pairs of programmers, sharing the same computer and keyboard.

Pair programming increases software quality without impacting time to deliver. Thanks to the improvement of the quality of their code, there will be less problems and bugs in the future.

Test cases are written before coding.

Creating a unit test\(^5\) helps a developer to really consider what needs to be done. Requirements are nailed down firmly by tests. There can be no misunderstanding when a specification is written in the form of executable code.

There is also immediate feedback while working. It is often not clear when a developer has finished all the necessary functionality. Scope creep can occur as extensions and error conditions are considered. If we create our unit tests first then we know when we are done. It is also important to run the test frequently so that we are aware of the problems that still are happening.

\(^5\) A developer writes a unit test that exposes either a software requirement or a defect. This test will fail because either the requirement isn't implemented yet, or because it intentionally exposes a defect in the existing code. Then, the developer writes the simplest code to make the test, along with other tests, pass.
Rules:

- Only write code for the current task. Never try to use the actual task time for developing code of a different task, even if the task are very related.
- Refactoring: Try to get rid of the duplication of code, make complex code simpler.
- Integrate often: Developers should be integrating and committing code into the repository every few hours, whenever possible. In any case never hold onto changes for more than a day. Continuous integration often avoids diverging or fragmented development efforts, where developers are not communicating with each other about what can be re-used, or what could be shared.
- Legible code.

![Diagram of the development process](image)

*Figure 7 Programmer estimates, customer chooses (Jeffries, o.a., 2000)*

**Test Driven Development TDD**

Test Driven Development (TDD) is a software development practice in which unit test cases are incrementally written prior to code implementation (An initial investigation of Test Driven Development in Industry, 2003).

With TDD, each new release begins with writing a test. To write a test, the developer must clearly understand the feature's specification and requirements. The developer accomplishes this through user stories that cover the requirements and exception conditions. This could also imply a variant, or modification of an existing test. This is a differentiating feature of TDD versus writing unit tests after the code is written: it makes the developer focus on the requirements before writing the code.
3.5. Lean, Scrum and XP

During the 90's Scrum appeared as a new concept to improve the efficiency in development teams. Due to its success, it was used by many companies but, despite the expected results, generating lots of problems. The majority of them were caused because the companies tried to follow the Scrum principles without the necessary adaptation to its company.

From our point of view it was necessary a framework in an upper level to assure this adaption, followed by a list of simple rules that can be easily remembered. That is, in our opinion, the main objective of Lean, make the company change its think, not only its process or methods.

Finally the hard work must be done by the developers. XP provides the developers the tools to work faster and team-working, which is very interesting for the empowerment of the people.

To summarize we can define the following interactions from the bottom to the top:

- **XP** is targeted towards improving the way of writing software. You cannot develop fast without it. Ideas like Test Driven Development, Pair Programming, Acceptance Tests, On-site Customer are the core tools that every team should use.

- **Scrum** is working in a higher level. Its practices can be applied to many different kinds of teams, not only software development teams. The aim is to increase
communication and decrease risk by helping stabilize the way to get wherever the team is trying to go.

- **Lean** is a concept that must be applied to the whole organization, use for reducing waste and adding value in every process. The idea is that every single process adds value to the product, in our case the software, and reduces the loss. Furthermore, the respect to the team, the commitment and the motivation of the staff are others of its targets.
4. Analysis – Problem identification

4.1. Survey Analysis

The following analysis has been conducted over a survey that was answered by the whole staff of the InnovationLab. The sample consisted of the six developers who constitute the developing task force of the organization.

A thorough analysis will be performed on the following chapters; the goal of this chapter is to show the results of the survey and to provide a simple summary of the responses.

Question 1. When do you document your work?

A first analysis of these results indicates that they are actually following the right way of writing the code. Lean Software Development (LSD) states that “If documentation will be needed, it should be done as the code is written” (4)

Question 2. How often do you perform unit tests?
There is something worrying about this regardless of the answer, and that is the fact that there is not a common way of working, exactly the same amount of developers perform the unit test, dairy, one a week or 2-3 times per week. There is not a set methodology to follow. LSD and Scrum states that test should be running as frequently as possible, if we intend to integrate the code every day, therefore there should be test before the integrations.

Question 3. Who test your modules?

A) You.  
B) Somebody else.  
C) You & somebody else.

There are two approaches for answering this question. The first one says that testing should be done by a different person rather than the developer. This leads to two ideas, that a person testing a code that is not his/her can find problems never conceive by the developers; the other idea is that the amount of time needed for that other person to develop the test will be longer than if the developer perform the test. Also we have to keep in mind that InnovationLab is seeking for a TDD (Test-Driven Development) methodology, which means only developers should only perform unit test in the implementing phase.

Therefore, we see that there is significant part of the developers who are being supported for the testing phase, something that Lean states not to be adequate.

Question 4. How often somebody test your own code?

A) 0-15%  
B) 15-25%  
C) 25-40%  
D) More than 40%

We believe that it is a problem that they release code that has not been going through a higher level test.

It seems that both unit testing and higher level testing is needed. Unit testing should be done by the developer. And the higher level testing “system test” should be done by someone else.
Question 5. How often do you create the unit test before coding?

A) Never.
B) Rarely.
C) Usually.
D) Always.

This question clearly states that the creation of tests is done mainly after coding, something that we are trying to get rid of.

Question 6. How often do you integrate your code?

A) Every day.
B) Once per week.
C) 2-3 times per week.
D) Less than once per week.

From these answers we clearly see that developers are currently working towards a very efficient and collaborative way of working. As Agile methodologies state, integration everyday is mandatory for a better rate of usability and reutilization. It also allows the developer to rapidly change tasks by leaving all the jobs on a day done, eliminating the waste of having partially done work.
Question 7. How many different tasks do you normally work during a day? (A task is the work done for a specific requirement in a specific work)

As stated by Poppendieck, “Software development requires a lot of deep concentrated thinking in order to get one’s arm around the existing complexity and correctly add the next piece of the puzzle.” (Poppendieck, 2006) It is therefore very important that the developers don’t switch from task to task very often, LSD states that a developer should only work on a task per day. We have to assume this is ideally since some task may take less than one day. Therefore we presume that answers A and B are correct but we see that 33.33% of the sample performs more than three tasks per day, and this is not acceptable.

Question 8. How much work-time do you spend daily in fixing problems or bugs?

With this question we aim to measure the quality of the released products, if they spend a lot of time in fixing bugs it means that there is not enough quality during the development process. On the contrary, the answers show that they are not having many difficulties to achieve projects with high quality since they don’t spend a significant amount of time in fixing problems and bugs. This might be due to the fact that the majority of the products that have been already released are not very complex and, therefore, they are less likely to have a high rate of bugs or problems.
Question 9. Who chooses the tasks you are going to be committed to?

One of the key things Scrum says about assigning tasks is that this should be done by the developers; each of them should be able to choose what they want to do in the next Scrum Sprint.

We believe there is a small necessity to move the process of assigning tasks to a more Scrum likely way.

Question 10. How often do you have direct contact with the customer?

By this question, we are trying to analyze the proximity of the customer to the development process. As Scrum and XP requires, customers and stakeholders should be very close to the development team, participating in the decisions made to continue to project, even the small ones. There is a lack of consensus in the answers; this is probably due to the fact that in many situations, the developers are also the people asking for requirements. Furthermore, some of the projects are so small that requirements done change and the goal of the project is easy to scope and follow.
Question 11. How often the projects you have worked on needed further improvement after they were released?

This is similar to question 8. The difference relies in that further improvement can mean fixing bugs or implementing new functionalities. The results make sense if we think about the size of the projects; the smaller a project is, the less likely it will need a change or a new functionality based on new requirements.

Question 12. How often do you have problems delivering software on time?

The goal of the question is to analyze the awareness of the efficiency in delivering software on time, to see if they realize about how often they do so. It seems that they don’t struggle in releasing the code on time. This might be caused by the project size; the underlying problem is “what will happen when projects starts growing?” The e-Me project has been in development over two years, and recently its requirements changed drastically, this means that the project has to start almost again and that it will not fit to the deadlines estimated.
Question 13. How many projects are you working on now?

More than an analysis question, this question helped us to know the distribution of work inside the organization. The fact that one third of the people are working in only one project and another third is working on three makes it hard to understand. Although the work load might be different, with more projects comes more task switching and, therefore, a waste arises.

Question 14. How often time estimations are achieved?

A first analysis of these results indicates that they not only usually achieve time estimations; it also means that they seem to be confident about their skills in estimating, but we need to compare these results with the ones from questions 8, 9 and 12.
Question 15. How much time do you spend per week in internal meetings?

This question is used to measure the amount of time they spend planning and doing follow up. For Scrum and XP standards, spending between 1 and 2 hours per week is not enough.

Question 16. Mark all the projects you are actively participating in:

The last question of the survey is designed to measure the size of all currently ongoing projects. With this information we will be able to better analyze the management side of the organization.
4.2. Finding wastes

Although there are seven principles, we performed the analysis based in the finding waste principle. We will use the other principles in the proposal since they give ideas on how to reduce and avoid creating waste.

Partially done work
The software development inventory is partially done work. The goal is to divide the work into small batches or iterations. We have identified five different types of partially done work at Innovation Lab:

- **Uncoded Documentation:**
  - This refers mainly to the requirements that were set at the beginning of the project. Projects change their requirements constantly, these requirements should be coded as soon as possible to avoid leaving them in a shelf and forgetting about them or realizing that they have changed and they are no longer useful.

- **Unsynchronized code:**
  - Workspaces and parallel development should be merged as soon as possible because the longer they stay separate the more likely problems will arise. Due to the size of the development team at the InnovationLab, there is not enough parallel development to be considered.

- **Untested code:**
  - Code must be integrated, tested and accepted; otherwise you are creating an inventory of partially done work. Writing code without a way to detect defects immediately leads to waste. At InnovationLab, there is not a fixed way of performing tests, we have seen in the survey that tests are done, with the same percentage of response, once per day, between two and three days, and in once a week. Lean Software Development suggests that tests should be done every time you finish a unit, without starting another task before it. Furthermore, it is better to have a homogeneous way of working, instead of three different ways of performing the tests, as seen on the survey.

- **Undocumented code:**
  - If documentation is needed, it should be written while coding, but only if necessary. At InnovationLab, we have seen that the majority of the documentation is written during the coding phase, although a 33.3% declares that they document after the whole program is finished. Lean recommends that technical writers should work parallel with developers in order to avoid writing unnecessary features. In our case of study, due to the size of the company, we believe this is an unnecessary aspect to consider.

- **Undeployed code:**
  - The releases should be done as soon as possible. Customers can learn more efficiently if the functionalities are deployed frequently rather than in big deploys that implies dramatic changes and more time to learn about them.
Extra features

- "If there isn't a clear and present economic need for the feature, it should not be developed" (Poppendieck, 2006)

- Some of the recommendations are to avoid creating a speculative application framework that can be configured to do almost anything; any feature that is not asked by the customer should not be added. During the e-Me workshops in which we have participated, we realized that, the idea of including many requirements in the early phases of development is a big source of waste.

Handoffs

- While working in development teams, one of the issues that arise is the need to hand off knowledge to colleagues. During the process, most of the knowledge (75% after two handoffs) is lost (Poppendieck, 2006). For this reason, companies should focus in reducing the handoffs. Some of the ways to do so are the use of cross-functional teams so that people can teach each other; the use of a fluent communication avoiding leaving tacit knowledge behind.

Task Switching:

- A main thing to consider is that developing requires deep concentrated thinking to solve the complexity of the tasks. As opposed to big companies where developers are more focused on certain tasks, in small companies the staff faces with many different tasks from different projects. This leads to a necessity to jump from task to task regularly.

- The effort in doing so is high and requires spending time in resetting their minds. At InnovationLab, one of the problems is the limitation of resources, as we have seen in the last question of the survey; all the developers are working in two or more projects at the same time. This leads to a waste of time when switching and preparing the mind for the members of the staff.

---

6 Tacit knowledge is knowledge that is difficult to transfer to another person by means of writing it down or verbalizing it (Collins, 2001)
Delays:

- A delay is the time waited for the information to reach its destination. Delays can appear while waiting for requirements, approvals, people assignation, people availability, code pass tests, etc.

- At the InnovationLab, delays are generated in some ways. Particularly, the e-Me project has a complex flow of gathering requirements. Due to the nature of the project, the requirements are generated by the development team itself, after this they are forwarded to the decision team to be approved. This leads to a waste of time while sending and receiving information. On the other hand, the InnovationLab relies many responsibilities on their developers, this causes to reduce delays while waiting for approvals or availability of a manager.

Defects

- Every code has to be tested to assure that covers all requirements efficiently. The classic methodologies try to solve these problems testing the code after finishing it. However, Agile methodologies recommend to move the testing to the beginning of the development and consider the results even in the design phases.

- One of the requirements of the InnovationLab was to implement a Test-Driven Development (TDD) methodology. As we have seen from the survey, unit tests are rarely or never created before coding. Writing unit tests before the code allows having simpler, more understandable and testable code.

4.3. Analysis of current methodologies

Throughout the process of gathering information, we have been able to not only find problems with the current way of working but we also found advantages that we believe they lead to a better lean philosophy. It is important to not only implement new work methodologies or concepts but also to retain, keep the ones that already exist.
Advantages

- **Horizontal communication**, due to the organization's characteristics, the employees have a fluent and constant communication not only between them but also with the manager and the administration board. "Members within an organization have an easier time with problem solving, information sharing across different work groups, and task coordination between departments or project teams" (Papa, 2008). Although InnovationLab is not a large company, an horizontal communication between developers working in several projects, speeds up the tracking process from the manager perspective and leaves more power of decision to the developers, trusting in their capabilities and make them feel more comfortable with their work.

- **Trust in employees skills and respecting the people**, leads to a better way of working. Employees at the InnovationLab are responsible to do their tasks in their own. Many decisions are directly delegated to the employees, which is at the same time motivating and empowering for the people in the performing of their job.

- **Interchangeable roles and tasks between the staff**, this implies two positive aspects; the first one affects to the knowledge of the employees. By working in different areas, employees acquire different knowledge; this is one of the core points in Lean, the continuous learning and people's flexibility. The second aspect refers to the increase of motivation, when developers change frequently of role or projects, they do not feel stuck in the same tedious job.

- **Partially implementation of Scrum**, this might be seen as a disadvantage because it makes people work in different ways. But we like to think that is an advantage since it will allow them to easily start using Scrum for all their projects, which is one the goals that we want to achieve.

- **Fixed release dates**, one of the things that Scrum suggests is that release dates should always occur on the same dates. This has two reasons. The first reason is that is easier to make time estimations when we always have the same time window. The second is that developers are more aware about the release dates and how much time and effort they have to put in their work to reach the end of a Scrum sprint.

- **Work alignment**, when employees are working on the same project they often try to work at the same time. This is a good principle because allows them to work closely and consult each other technical issues. Also, it is a good way to avoid handoffs waste.

Disadvantages

- **The e-Me project disadvantages**: We want to focus in this project because is the most important for the organization, is the most ambitious and complex too.

  - Workshops: We have seen several problems during these meetings: there was no clear goal and no clear idea about the structure of the meeting, there was no moderator to control the rhythm of the meeting, no decisions were made, the topic of the conversation was changing constantly without any conclusions
done. After several hours there was no clear conclusion. Many things were trying to be achieved but none were actually.

- **Requirements:** Due to the nature of the project there is not a typical customer requirements, the responsibility in creating the requirements relies in the developers who must define the requirements and then forward them to the responsible of the project who must accept them. There is a lot of time spent in this process. Furthermore, there is no real concern about which functionalities are the most important for the project.

- **Role definition:** During the project’s lifetime, there has been changes in the people assigned and the roles they performed. This, which Lean states that usually is good, can lead to a misunderstanding about responsibilities and duties of the people involved.

- **Main functionalities:** Lean Software Development claims that requirements should be narrowed at the beginning, paying attention in the very clear ones, which add value to the product. Therefore, there should be a clear goal of the functionalities of the first release. In the e-Me we saw that functionalities for the first release are not set leading to a constant flow of new functionalities without prioritizing them. Lean states that the project should focus in delivering soon the core of the system with the functionalities that will be most used.

- **Meetings:** Meetings should always try to focus, as daily Scrum meetings, in three questions every developer should answer:
  - What have you done before?
  - What are you going to do next?
  - What obstacles are there in the way?

  Although this set of questions is formulated for the daily Scrum meetings, they can also be very useful for more general meetings. The goal is to speed up the development process by focusing in what is really necessary. InnovationLab team spends between two and three hours per week in meetings (see question #15 in the survey). Considering that in this time they are having: general meetings for the regular situation of the organization, project meetings (for each of them) and daily Scrum meetings we believe this time should be reduced to spend between one and two hours.

  One of the ways to do so, is to establish a visible structure for the meetings, in which every single person knows in advance what the meeting content will be and knows what they have to bring to the meeting. Another way is to provide the manager with a tool to keep track of the status of every project and to know what will be done during the following days.

- **Tasks switching:** InnovationLab’s staff has to deal with many projects, but the number of employees is, obviously, limited. The company has the necessity to develop all of them on time. When a person has to switch between different tasks his/her performance is lower than whether he/she is focus in only one. In the survey we can see that a 66% of the employees deal with more than two tasks per day.
- **Scrum is not fully implemented:** the roles of a scrum team are not completely defined. For example, there is no product owner in e-Me and in the rest of the projects the number of people involved is one or two; this means, that there is one person that has to assume all the roles and this contributes to lack of efficiency in his/her work. The manager tried to encourage the employees to use a Product Backlog, but right now is not being used by anyone.

### 4.4. Problems when globally implementing Scrum and XP

In this section we want to focus in the possible problems that may arise while adopting Scrum and XP as the main methodology framework to work with.

We have to make the effort to analyze the possible implications of adopting these methods in the InnovationLab. It is known that Scrum and XP are a set of general rules, therefore is not possible to suit those rules and methods in every type of organization. There is a need to better understand the ideas that drive the methodologies to create a customized framework for every single organization or project.

We get this idea when we see how, in the literature, they talk about "the number of members in a team should be between 5 and 9" or "the sprint should be 30 days" (Softhouse consulting, 2010), this gives the idea that the methodology is open and it is up to the team to create their personal, customized framework.

Throughout this section, we will only depict the problems we have been able to identify so far, the way to deal with them will be presented in the following chapter.

We have identified two main categories for the most likely problems we will face during the implementation: Problems concerning project sizes and problems concerning InnovationLab size. We discuss about them in the next section.
**Problems concerning project sizes:**

- Time spent setting up small projects: InnovationLab has a large variety of projects if we consider the size of the organization. Many of these projects only require a dedication of less than one week. The cost of setting up projects like these from the Scrum point of view is very big. There is no need to assign a Product Owner or a Scrum Master, there will not be a Scrum team but a single person will take care of it. So, therefore, how do we deal with this? We will have to provide a framework that can efficiently deal with the managing issues of these projects.

- Product backlogs: If a single person is working in up to three or four projects, why he or she should check three or four different product backlogs?. How can a single person (the manager) follow the status of six or seven different product backlogs?. The complexity of these issues makes Scrum hard to implement in its standard basis, a new approach needs to be considered.

**Problems concerning InnovationLab size:**

- Scrum and XP talks about small team sizes between five and nine. They also recommend that teams should have sufficient skills within to get the job done. This implies that the team has the requisite testing skills, database skills, user interface skills, apart from the core development skills. However, InnovationLab can impossibly create teams that big. If we take a look at question number sixteen in the survey, we see how the majority of the projects are being developed by an average of two developers.

- Other roles besides Scrum Team:
  - On the previous point, we have seen how projects are handled mainly by two developers. From the Scrum perspective, there are more people involved in a project, the Scrum Master (SM) and the Product Owner (PO). How do we deal with this issue in a small organization like InnovationLab? Having one SM and one PO for every single project can lead to have 9*2 = 18 roles. This can obviously not be considered. Shall we then consider combining both roles?

  - The Product owner is responsible for concepts and ideas (i.e. the backlog), while Scrum Master is responsible for execution and quality, so the Product Owner wants more features while Scrum Master is focusing on getting it done. Performing both tasks can be very difficult to handle by a single person because the goals of the roles are different and crash.
5. Proposal for implementation

5.1. Lean implementation

In the book "Implementing Lean Software Development: From concept to cash" (Poppendieck, 2006), a 21-step program for implementing Lean Software Development is proposed. This proposal is a suggestion of what you may have to do. Adapting this structure we will try to customize the suggestions provided to create a customized set of steps to follow.

We must underline that due to the size of the development team there will not be an outstanding improvement in the overall efficiency of the InnovationLab. This is mainly because Lean is intended to be used in companies that have a certain size, companies with various development teams in which the efficiency has been decreasing while the number of employees was increasing. Lean principles help companies to eliminate the waste generated in the interaction between different stakeholders of the company and also customers.

People

Our first gaze is in the people inside InnovationLab. We strive to analyze they way in which they work together, collaborate and work as a team. The first and most important thing to underline is people's knowledge in Lean principles:

- There is a need to train all the members of the InnovationLab in Lean. People must be aware of the best practices of Lean Software Development so that they can start using them in their daily work. Some awareness posters will be provided to the members of the staff with the goal of hanging them in their place of work. We also strongly recommend a short training period for the members that would consist in two or three sessions separated by a month or two. This structure will allow the staff to retain the main principles of Lean.

Secondly, InnovationLab must keep responsibility and decisions by the level of developers.

- From our interviews with the manager, we have learnt that the organization is looking forward to growing in two aspects, by hiring a new developer and by merging themselves with other IT departments in the University. When organizations grow in the number of employees the management effort is frequently multiplied. It is a tendency to implement mechanisms to keep track of the work done by the employees or to create intermediate management roles. Companies should be careful when applying these tendencies.

- InnovationLab is currently working in a very lean environment when it comes to responsibilities and decision levels. They must put an effort in trying to keep this way of
working even if the number of employees grows suddenly.

**Deliver Fast**
Delivering fast is the act to be able to release usable services as soon as possible. Fixing the release dates to every three week or four makes the team to have a reliable and repeatable cadence of work. The goal is to establish a predictable level of work that can be done. This date should be the same for all the team, doing so, the team is aligned and committed to a common goal, even if they are working in different projects.

Choosing tasks, pull scheduling:

Team members should only select only those tasks they are confident that they can complete. From the survey we have identify the lack of consensus in this matter, realizing that only a 16.67% of the developers chooses by themselves. "During the first couple of iterations, they might guess wrong and select too much work. But soon they establish a team velocity, giving them the information they need to select only what is reasonable." (Poppendieck, 2006)

Three week iterations. Although this is also commented in the following chapter 5.2, a three week iteration makes the team have a releasing date fixed, allowing them to have something deployable very often.

**Defer commitment**
InnovationLab should abolish the notion that is good to start a development with a full description of the requirements. We have identified from the interviews and the workshops we have participated in that very often they try to set a full list of functionalities and requirements. These lists are sometimes depicted and described in depth creating a lot of documentation that is likely to change in the future, making the previous work useless.

A similar problem happens during meetings and workshops. Discussions during workshops should only be focused on how to break dependencies in the architecture and talking about the most prioritized requirements to be built, the rest should not even be considered. A goal needs to be fixed before the workshops, creating atomic functionalities that can be created without dependencies with other functionalities, modules, etc.

**Eliminate waste**
We will assume the reader has read the previous chapters of this report and, therefore, we will go straight to recommend the principles to follow:

- Do not write long term requirement's documentation. Avoid trying to elaborate a rich and detailed document of requirements, try to be concise. Get a list of the main requirements and write them in the product backlog. For every sprint InnovationLab will start, contact the customer to get a detailed description of the requirements that are going to be developed in that iteration.

- Testing efforts should always be done by the responsible for that piece of code. When the rest of the lean principles are well understood, there is no need to perform the test
from another point of view that is, another person or team, because the developer will be more committed to the completion of the task and he will have the exact idea of what the customer need.

- Write documentation while coding, the longer you wait the more time you will have to spend writing the documents. Document as soon as the tests have succeeded.

- Eliminate extra features: While planning the next iterations and the stories (tasks) to develop, always prioritize the most demanded, the ones that are most valuable for the customer and the ones that will be most used. Keep this principle always in mind, forget about the other tasks even in the meetings.

- Task switching can be one of the biggest sources of waste in the company. The eagerness of InnovationLab to acquire as much projects as possible is leading them to increment the complexity of the project management. It is making developers to have to deal with many different projects and, therefore, tasks with different scope. To avoid this problem, InnovationLab must make the effort to limit the number of tasks performed per day or week, or at least make the developers aware of the decrease in productivity that this implies. It is also highly recommended to set up a working effort percentage for the most important projects. Limiting the amount of small projects that the InnovationLab can acquire per iteration.

**Training**
In order to implement a Lean philosophy in an organization, at least one training session should be performed. The idea is that since Lean is a set of principles and not only rules, the better understanding the team has about it the more likely we will see benefits. It is very important that every single member have a clear idea of what are the principles to follow and that they should keep them in mind all the time.
The training will be supported by the awareness posters; they will reinforce the principles in the employees' minds.
5.2. Framework

The idea of our proposal is based in the use of three different methodologies: Lean, Scrum and XP. As you can read in Chapter 3.4 these methodologies are compatible between them and provide a good start point for any company willing to improve its performance.

![Framework Diagram]

### Structure

Due the limitation of the staff and the number of projects, Scrum cannot be directly applied. From our point of view it is impossible to maintain one product backlog for each project. Because of this, InnovationLab should apply the following steps:

- Create a common Scrum Sprint that should include all the tasks of the running projects. Because:
  - It is impossible to maintain the roles for every project.
  - There are more projects than developers.
  - The size of some projects is very small.
  - Having a common meeting for the whole team allows the developers to know the status of the other projects; this is useful in case of project switching (assigning developers to other projects). This creates a more flexible team, it allows developers to easily move from project to project and also participate in the discussions of the other projects.

- Have two "Weekly Scrum" meetings with some fixed points that will have to be answered by the developers, following Scrum methods (Chapter 3):
  - Mondays and Wednesdays.
  - The developer will inform about the complete tasks.
  - The problems during the week.
  - The next tasks involved.
  - Update the burndown chart (read planning and management tools section).
Gather all the information from the projects, taking the role of the Product Owner only in the medium or large projects. For the small projects the Product Owner will be the developer and will inform him. Because:
- He/she can pay more attention to the projects that are more important for the InnovationLab.
- The developer can learn another role and feel more useful.
- He/she will have the total view and control over the amount of tasks that are pending for the company and their priority.
- He/she will assure the correct distribution of the tasks between the developers, during the Planning Meeting.

Fix a moderator role for every meeting, before starting, and follow some simple rules:
- Follow up a concrete way depending on the type of meeting.
- Make everybody in the meeting participate and listen to everyone’s opinions.
- Write down the most important ideas or conclusions.
- Finish the meeting with a high level of commitment about the topics discussed and conclusions obtained, making a summary and reading it.

Fix the duration of the Scrum Sprint in 3 weeks. Because:
- If the time is shorter, there are too much meetings in a short period of time. This, from the Lean perspective can be seen as a waste. However, if the sprint is longer the capacity of reaction to fit small projects in the iterations will be reduced.
- The manager will have enough time to prepare and decide what are the most important projects and tasks to do in the following Sprint.

Testing

We propose to follow the philosophy of Test-Driven-Development (TDD) because it fits well in Lean and Scrum Perspectives. For this reason the tests should be written before coding. We
found out that the company was using TDD in some projects. We strongly recommend implementing the following principles (Beck, 2002):

- Start each feature writing a test. It will not be recommended to add any functionality without writing the test before.
- Check whether actually the test fail (It must fail, because the functionality is not developed yet)
- Write the code necessary, only, to pass the test. Do not add more functionality.
- Run all tests and check if the code works correctly.
- Clean up the code and re-factory it, trying to avoiding the repetition and generating an elegant piece of code.
- Repeat the cycle.

Other good ideas for the company can be extracted from Behavior-Driven-Development (BDD). This technique is focused on exposing the logic to the stakeholders using natural language. Describing, then, the benefits a purpose of the code. Ideas such as (North, 2010):

- Test methods names should be sentences.
- Every class should do something; it is useful to define one test for the current class. Naming the class with this word the functionality is restricted and to clarify the responsibility about the behavior.
- Use an expressive test name. It will be useful when the test fails.
- Determine the next more important behavior asking this question: What's the next most important thing the system does not do?
5.3. Planning and management tools

On the above figure we have created an approximate chart that explains how InnovationLab's scheduling is currently managed. You can see the full figure in Appendix B. There is a timeline with five different regular projects and six small projects in which the work developed is divided between the different Resources R#. For the regular projects there is usually more than one resource while in the small projects there is only one resource assigned.

Thanks to this representation, we can see how complex is to schedule the tasks and work that the different employees have to perform when there are a considerable amount of projects. The difficulty lies in not only assigning too much workload from different projects to the developers but also to be able to make them work in the same projects as often as possible. By not having an efficient tool for planning and scheduling (they currently rely on an Excel sheet) the risk of not performing an appropriate planning are increased.

As part of the planning and following up of the tasks committed (with an invoicing goal in mind), InnovationLab uses another tool besides the Excel sheet called InTime. InTime is used by the employees for inputting the amount of time dedicated to each project, stating the type of work developed and the name of the task. This functionality, if merged with the Excel sheet, could be used as a measurement tool, to compare the estimated times with the actual times. This information could also be used to sharpen the estimations and improve the accuracy of the project planning and budget.
There is also a SharePoint server in which each project is managed separately. There you can see a list of the remaining tasks, their priority, documentation, etc.

And what do these three services have in common? They all share the tasks of the projects so, why not creating a single and centralized tool that can managed the three functionalities?

To avoid these problems and to ease the management tasks, we want to propose a tool that can merge the functionalities of the different tools currently used and provide a solid structure to start working in an Agile environment. There are obviously many solutions in the market that can fit better or worse in the requirements we need. We want to focus our proposal in a certain commercial product. We have tried different commercial solutions that are currently being offered but we believe this is the one that better suits our approach. Our proposal does not recommend specifically the purchase of this solution; we just want to give a practical example. It is up to InnovationLab to perform further trials and test to decide what they want to do about it. The solution proposed is called OnTime® 2010 from Axosoft.

This tool will allow the manager to better keep track of all the current projects but, most importantly, will ease the planning effort for long term and for every Sprint. On the main interface we can see how we can list all the feature stories that are in the backlog, it allows the administrator to sort the features by priority (previously assigned in consensus with the customer or the product owner) and by project.

Figure 13 Main Interface. OnTime® 2010 from Axosoft.
With the same interface we can also control the Sprints of each project or a general Sprint grouping tasks from different projects. Please note the column on the right side of the interface called “Project”.

Figure 14 Sprints Management. OnTime® 2010 from Axosoft.

Another functionality that can drastically improve efficiency in the InnovationLab is the “Customer Portal”. A portal for customers in which they can check the current status of their project and input new requirements, bugs or tasks (always depending on the rights assigned to the customer). Due to the nature of the majority of customer relationships at InnovationLab, a tool like this can allow the different stakeholders to participate actively on the development process, creating a feeling of proximity and making them see how the product evolves through time. This also can help in the e-Me project, for instance, by sorting the gather of requirements process described previously in this report.

Figure 15 Customer Portal. OnTime® 2010 from Axosoft.
**Burndown chart**

One of the recommendations is to use a Burndown Chart. This tool provides to the manager a very interesting planning view. At the beginning of each project estimation should be done about the number of hours that the project will need.

In each “Weekly Scrum” the team should create its own chart in order to compare the estimation with the actual work done. This information will be very useful in future projects estimations, although, in InnovationLab this tool should be only used in the largest projects such as e-Me and TFU.

The Y-Axis corresponds to the effort (in hours) and the X-Axis corresponds with the “Weekly meetings”

![Figure 16 Burndown example](image-url)
5.4. Awareness posters

The intention of using awareness posters in InnovationLab is to make the staff know more about the methodologies, and to think differently about the things that they may be learning or doing. From our point of view, we recommend to use them as a tool to motivate and encourage the people about the new methodologies.

Figure 17 Poster-1. Software Developer / Analyst Duties

Figure 18 Poster-2. Test Driven Development steps
Figure 19 Poster-3. Surviving to the change. Inspired by (Johnson, 1998)

Awareness posters can be a help to remember some important ideas. The idea is to keep them for a time in the company in order to evaluate its effects. If they seem useful for the team we recommend keep them otherwise, after a period of time, should be removed.
6. Future Research

During this study, two main concepts or objects have been proposed. The first one, in which the second one is based, is the proposed framework that combines Lean, Scrum and XP. This framework has been used due to the needs of the InnovationLab to implement new working standards. The framework has not been fully studied for target organizations different than the InnovationLab; there is not a definition of how to combine the three methodologies into one general methodology that can be used for different organizations. Therefore, our first suggestion for future research is the need to follow the study of this research and to create a methodology that can be adapted to other software development organizations independently of their size or work nature.

Another way to follow the studies conducted in this research is by analyzing the proposal for implementation. This proposal for implementation can be used as a starting approach to create a theoretical framework for different organizations with similarities to InnovationLab.

Lastly, the authors would like to analyze the way in which the implementation of their proposal affected the development of the InnovationLab. A research could be conducted to analyze how the organization changed and evolved by using different key Performance Indicators or KPI. Although this might be seen as a practical research, it can be approached by creating an analysis framework for methodologies implementation in software development companies.
7. Conclusions

After deeply studying the company, learning of lean-agile methodologies, the information gathering, analysis of the problems and the compilation of the proposal, we conclude that:

- About InnovationLab:
  - The company is in a good situation from the customers' point of view, due to the partial implementation of Agile techniques. However, a total implementation is desirable.
  - The manager will have more control and more information about the situation of the company. An important point if we consider the future growth of the company.

- About Lean Software Development:
  - It is a good philosophy to apply to any IT-company, independently of the working methodologies used.
  - It really suits with Scrum and XP practices, due to its principles.
  - The seven lean-principles should be known by every person in the teams, in order to follow them.
  - The cost of implementation is not expensive, but requires a personal effort of every person of the staff to assimilate the concepts.

- About the framework (Lean-Scrum-XP):
  - Scrum meetings and pull systems helps the company to reduce waste and focus in the work to add value for the customer.
  - Good programming practices, pair programming and Test Driven Development assure a good quality in the development level, creating value in every action.
  - One of the duties of the ScrumMaster role is to teach the team and make the people learn from his/her experience; this is a plus to empower the commitment.
Appendix A – Interview transcripts

Interview questions to the developers

Role in the organization:

- What methodology does the Innovation Lab uses nowadays? (to see the if they are aware of it)
- How often do you help your colleges in programming issues? (Is XP concept somehow used?)
- How often do you write documentation? When do you do it? Not so sure about this question…
- Means of communication with the customer?
- What is the procedure to follow when the requisites change?
- Do you follow any pattern when assigning priorities to tasks to do?
- How do you plan the testing stages? By date, by finished module, according to the rest of the team?
- Who tests your modules?
- What do you think about the software development method used by the company? How would you improve it?
- Do you find useful the tools that you are using nowadays? Are they enough?
- Do you think that the amount of documentation is enough for each project?
- Do you find difficult to follow some college’s code?
- Do you find difficult to follow your code after a while?
- What is your opinion about the communication methods used currently?
- Is vertical communication delaying processes?
- Are you motivated by the company to do your work? Do you think necessary more personal development?
Interview questions to the Manager

Number of employees in the organization:

- Roles in the organization (although we have asked this before, it would be interesting to see if they actually answer different things, this could show lack of consensus or hierarchy)
- What kind of software development method that the company is using currently? How it works?
- What kind of tools are you using? Do the people find them useful?
- Is the current documentation comprehensive?
- What is the aim of the company? I.e. give high priority to satisfying customer through early and continuous delivery of valuable software
- Are changing requirements allowed? Even during late stages of development?
- How are changes through development faced?
- What is the period to deliver software? I.e. could be short, from a couple of weeks to a couple of months.
- What is the communication method used during the software development? Inside the company?
- How is the company measuring the progression of the work?
- What types of developers are in the company? Are they self-organizers, can be re-organized continuously in different configurations to meet the requirement of the new projects?
- Does the team have in each project meeting to reflect how to become more effective then tunes and adjusts them?
- Highlights of the process
Interview transcript with the Manager

- All developers work in more than one project.

- Around 7 projects in parallel.

- Projects are from 2 days to 3 years.

- Big projects like e-me or TFU.

- TFU is for managing teachers hours and time from the university perspective, keep them controlled.

- They have open-source (because are researching projects) and proprietary software.

- More difficult in open-source because there is not a clear set of requirements.

- Methodologies: SCRUM, tried a little bit of XP but not strictly, others.

- e-me: SCRUM
  Meeting every two days.
  H. is ScrumMaster.
  M. L. and O. are the managers, they decide the requirements.
  The functionalities are sometimes offered by the developers instead of the managers, they have to accept them after.
  Every sprint starts with brainstorming because of the lack of requirements.
  The deadline for the e-me 2.1 is in autumn.
  They changed the scope of e-me after one year of working on it, because they didn’t have a clear goal of what they wanted and the lack of requisites.
  4 people work for e-me.
  They decided that 2.1 will be focused on social networks such as facebook or tweeter.
  They now want to use the e-me not only for students but also use it for bringing together small scattered communities throughout the north of Sweden.

- Documentation: They write it but is not formalized, different structures, different appearances.

- H. used to work in Capgemini. They use to use RUP for documentation with a lot of templates.

- Test-driven development. He thinks it could be interesting.

CONCLUSIONS

- They are open to propositions, he is interested in Lean software and Scrum but talks about using the 3 of them.
- For him is very important the documentation and wants to put an effort on it.

- It seems like Lean software Development is a set of ideas or hints for people to consider in their work regular basis. It is also a set of rules to follow when you already have a defined work methodology, in our case, we might have to set up the right methodology and start from scratch.

- They are expecting to grow in number of employees.

**Interview transcript with the developer-1**

- He was a former scrum master for the e-me project.

- **SCRUM**
  Positive: Collaboration, the master can follow up very close and now how the developing is going on, can easily adapt to change.

  Negative: SCRUM has a rudimentary planning that makes difficult to asset the budget, time and resources required, it makes de development have stressing peaks of workloads.

- Working right now in 2 projects, e-me and public transportation.

- When it comes to work on e-me they all agree on working the same days, that allows them to work more focused and to collaborate.

- They don’t have any methods for prioritizing the tasks.

- There is a lot of collaboration between the developers as in XP.

- They use SharePoint.

- No procedures when they spot and actual or future problem.

- No specific method on how tasks should be divided, they put them all in the wall and each dev. chooses the ones he/she likes.

- Testing: They do unit testing, lack of knowing when it should be done

- For e-me they have a test manager role
Interview transcript with the developer-2

She is currently working with 3 different projects:

- TFU: Project for planning teachers schedules and hours
  Customer: University of Boras, trying to sell it to other universities
  ASP.Net
  Started in Autumn 2007
  They don’t know the deadline
  Requirements changing constantly
  Methodology XP
  Currently 3 developers are working on the project.
  Boras is always forcing them to release as soon as the code is ready, they can’t apply Scrum.

- VFY: Tool for managing the students internships
  Customer: University of Boras
  ASP.Net
  Started spring 2007
  Finished in late 2007
  Started using January 2008
  Methodology XP
  Currently two developers working

- QlikView: Business Intelligence software for Financial management of the college
  She works alone with this.
  It uses its own scripting language.

Development tools: Visual Studio, SQL, Sharepoint, Sourcesafe and Visio

For testing: They started using TDD (Test Driven Development) but over time they were not being strict and eventually they abandon it. Anna liked and would be willing to use it again, though she complains about complexity

They have general meetings every Monday : I asked why on Mondays, she says is good for preparing the weekend, I said is bad if we want to analyze the problems faced during the previous week. She was agreeing with it.

For the projects involving more than one person, the team has a meeting every week too.

They always try to have the same dates for releasing to production.
Appendix B – Approximation of current task scheduling.

<table>
<thead>
<tr>
<th>Project 1</th>
<th>Project 1</th>
<th>Project 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>R1</td>
<td>R1</td>
<td>R1</td>
</tr>
<tr>
<td>R2</td>
<td>R2</td>
<td>R2</td>
</tr>
<tr>
<td>R3</td>
<td>R3</td>
<td>R3</td>
</tr>
<tr>
<td>R4</td>
<td>R4</td>
<td>R4</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Project 2</th>
<th>Project 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>R1</td>
<td>R1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Project 3</th>
<th>Project 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>R1</td>
<td>R1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Project 4</th>
<th>Project 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>R1</td>
<td>R1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Project 5</th>
<th>Project 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>R1</td>
<td>R1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SmallProject1</th>
<th>SmallProject1</th>
</tr>
</thead>
<tbody>
<tr>
<td>R1</td>
<td>R1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SmallProject2</th>
<th>SmallProject2</th>
</tr>
</thead>
<tbody>
<tr>
<td>R2</td>
<td>R2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SmallProject3</th>
<th>SmallProject3</th>
</tr>
</thead>
<tbody>
<tr>
<td>R3</td>
<td>R3</td>
</tr>
</tbody>
</table>

Task switching

Time
<table>
<thead>
<tr>
<th>Project 1</th>
<th>Project 2</th>
<th>Project 3</th>
<th>Project 4</th>
<th>Project 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>R1 R1 R1 R1 R1 R1 R1</td>
<td>R2 R2 R2 R2 R2 R2 R2</td>
<td>R1 R1 R1 R1</td>
<td>R3 R3 R3 R3 R3 R3 R3</td>
<td>R5 R5 R5 R5 R5 R5 R5</td>
</tr>
<tr>
<td>R2 R2 R2 R2 R2 R2 R2</td>
<td>R3 R3 R3 R3 R3 R3 R3</td>
<td>R2 R2 R2 R2 R2 R2 R2</td>
<td>R3 R3 R3 R3 R3 R3 R3</td>
<td>R1 R1 R1 R1 R1 R1 R1</td>
</tr>
<tr>
<td>R3 R3 R3 R3 R3 R3 R3</td>
<td>R4 R4 R4 R4 R4 R4 R4</td>
<td>R4 R4 R4 R4 R4 R4 R4</td>
<td>R5 R5 R5 R5 R5 R5 R5</td>
<td>R6 R6 R6 R6 R6 R6 R6</td>
</tr>
<tr>
<td>R4 R4 R4 R4 R4 R4 R4</td>
<td>R5 R5 R5 R5 R5 R5 R5</td>
<td>R5 R5 R5 R5 R5 R5 R5</td>
<td>R6 R6 R6 R6 R6 R6 R6</td>
<td>R7 R7 R7 R7 R7 R7 R7</td>
</tr>
</tbody>
</table>

SmallProject2

SmallProject5

ct4

R5

R4 R4 R4 R4 R4 R4 R4 R4 R4
References


