CHALMERS

Thinking Lean in Agile Software Development Projects

A qualitative study to Identify ways to improve productivity and increase business value

Master of Science Thesis in the Master's Programme International Project Management

BRISILDA KOLA

Department of Civil and Environmental Engineering
Division of Construction Management
CHALMERS UNIVERSITY OF TECHNOLOGY
Göteborg, Sweden 2014
Master's Thesis 2014:119

MASTER'S THESIS 2014:119

Thinking lean in agile software development projects

A qualitative study to identify ways to improve productivity and increase business value

Master of Science Thesis in the Master's Programme International Project Management

BRISILDA KOLA

Department of Civil and Environmental Engineering
Division of Construction Management
CHALMERS UNIVERSITY OF TECHNOLOGY
Göteborg, Sweden 2014

This thesis is dedicated in loving memory of my father,

Ndue Kola (1958 – 1994).

Thinking lean in agile software development projects

A Qualitative Study To Identify Factors That Can to improve productivity and increase business value

Master of Science Thesis in the Master's Programme International Project Management

BRISILDA KOLA

Technical report no 2014:119
Department of Civil and Environmental Engineering
Division of Construction Management
Chalmers University of Technology
SE-412 96 Göteborg
Sweden
Telephone + 46 (0)31-772 1000

Department of Civil and Environmental Engineering Göteborg, Sweden 2014

ACKNOWLEDGEMENTS

I would like to express my special gratitude to all those who supported me during my studies, and especially during this project.

My deepest appreciation goes to the Swedish Institute for having provided me this precious opportunity to pursue my master studies in Sweden, by awarding me a scholarship.

I would like to acknowledge with much appreciation the role of my supervisors Claudio Benghi and Christian Koch for the valuable comments, guidance and advice.

I would like to thank all the participants of my study, for willingly sharing with me their precious time and experience, and for giving me the permission to use and publish their names and the information that they provided for the purpose of this master thesis. Thank you Mattias Gustafsson, Stefan Singman, Magnus Jonsson, Andreas Sandberg, Vincent Thavonekham, NK Shrivastava, Liang Zhang, Paul Oldfield, Steve Fenton, Tim Vermeerschen and Johnny Herrmann.

A special thanks goes to my family for all the understanding, the unconditional support and the endless love provided day by day.

Last but not least, many thanks go to my relatives and friends, especially to my best friend Adelina, who made my life easier just by being there, helping and supporting me all the way.

Thinking lean in agile software development projects
A Qualitative Study To Identify Factors That Can to improve productivity and increase business value

Master of Science Thesis in the Master's Programme International Project Management

BRISILDA KOLA
Department of Civil and Environmental Engineering
Division of Construction Management
Chalmers University of Technology

ABSTRACT

Background – The high rate failure of software development projects, which influences directly the business value of the organizations in which they take place by decreasing the revenues and affecting employees' productivity, is the main motivation to conduct this research.

Objective – The objective of this study is to find appropriate ways to measure and improve the productivity of the teams and to discover to what extent the agile methods, the integration of the lean principles and the "soft skills" of the team can be combined in order to achieve high quality results.

Method – A qualitative study, consisting in interviews with only Scrum masters and online questionnaires with both Scrum masters and practitioners, was done in order to observe to what extent the views of the scholars and researchers, whose work is used to build the theoretical framework of this dissertation, apply to real life organizations operating in software development industry.

Results –Scrum masters and practitioners have found several ways to measure the productivity of the teams, based on their working experiences. Agile and lean methodologies have proved to be beneficial to software developing organizations; while "leagile paradigm" remains a relatively new concept in this industry. To conclude, people are the most important asset in every organization, so their superiority is recognized and worshiped by improving the management practices.

Conclusions – The integration of both agile and lean methodologies has a positive effect on software development project-driven organizations. Nevertheless, the process is often associated with challenges that can affect the productivity of the teams. Therefore, it is

worth to find an appropriate way to measure productivity while continuously considering and improving the “people skills”.

Keywords: software development, productivity, agile, Scrum, sprint, lean, Kanban, leagile, management

Table of Contents

Table of Figures

List of Tables

PREFACE

The author of this research finds the motivation in her personal interest in IT industry, more specifically in analyzing and finding ways to improve the work processes in this domain. The rationale for this study was found in the previous research, which shows that software projects often fail to fall in the iron triangle of time, cost and quality. This affects noticeably the business value of the respective organizations and their position in a strongly concurrent market.

This study aims to connect the productivity of the project team with the business value of the organization that runs the project, to find ways to measure this productivity and to observe whether agile and lean approaches lead to productivity enhancement. This is firstly done by looking into previous scholars' research in order to build a theoretical framework that aligns with the current study and support/contradict its findings. The findings are drawn by conducting interviews and creating questionnaires submitted by people experienced in developing software in real life organizations.

Afterwards, the collected results are observed and analyzed, with the intention to find ways to measure and improve the productivity of the team; to understand the impact of integrating different software development approaches in the work practices; and to realize the vitality of good management practices throughout the whole process.

By the end, the reader is provided with a summary of best practices and suggestions on how to increase productivity and enhance business value.

1. INTRODUCTION

This chapter will represent the reader with the rationale for the current study, its main aim and objectives, the methodology, and the layout of the research. The introduction chapter ends in a short description of the current work's limitations.

1.1. Rationale for the study

Previous research shows that the trend towards a customer-centric tendency of the companies is very clear in the current economic situation all over the world. In a volatile marketplace, the organizations should be prepared to handle and respond to the changeable and complex customer requirements, personnel, cost and schedule (Mishra and Mishra, 2011). This flexibility is provided by integrating agile approaches in software development processes. Indeed, operating agile means to be able to rapidly and inherently create, respond and embrace change in business and technical areas (Highsmith and Cockburn, 2001; Dingsøyr, *et al.*, 2012). Agile approaches encourage the developers to learn from experience and add to customer values by reducing cost, improving quality and maintaining simplicity, while deemphasizing long-term planning in favor of short-term adaptiveness (Dinakar, 2009). In other words, agile approaches enable the team to incorporate effectively the user requirement changes during the life cycle of a project (Lee and Xia, 2010). They have proved to increase the productivity and predictability of software development projects, while decreasing their defect rates (Grenning, 2007).

All these advantages associated with the usage of agile approaches, stimulate the author of this thesis to conduct research on how these methodologies impact the productivity of the people working with them and on what can be done to improve the productivity, add to business value and raise the organization's returns on investments. Several scholars have conducted research on finding ways to increase productivity of the teams, but still no ideal measure has been found (Maxwell and Forselius, 2000) so the productivity of the teams remains a field to explore.

Other researchers see agility as a concept closely related to leanness (Naylor, *et al.*, 1999; Mason-Jones, *et al.*, 2000a, 2000b; Eaton, 2003; Agarwal, 2006; Goldsby, *et al.* 2006; Krishnamurthy and Yauch, 2007; Naim, 2011; Soni and Kodali, 2012; Wang, *et al.*,2012; Vinodh and Aravindraj, 2013). The aim of leanness is to eliminate waste, reduce cost and increase business value, while creating effective responses and business outcomes

(Agarwal, *et al.*, 2006; cited in Dingsøyr, *et al.*, 2012). Dingsøyr, *et al.* (2012) also claim that a combination of agile practices and lean ones has proved to be efficacious, but yet empirical validation of such assertions is missing. Taking into account the advantages of both methodologies, this study aims to investigate the benefits that a potential combination of these methodologies would bring to software development project-driven organizations.

Finally yet importantly, a huge number of academic articles and managerial books observe the importance of the human factor in the whole software development process, so this thesis aims to find out what should project managers take into account when aiming to achieve a high organizational success.

1.2. Aim and objectives

The main goal of this master thesis is to investigate how can agile and lean methodologies help towards enhancing productivity in software development projects. This is achieved by firstly creating a supporting theoretical framework using relevant information from previous research and literature sources. Afterwards a concrete research is conducted by using the experience of the participants involved in this study to answer a number of research questions shown in figure 1.1. The main objectives set in order to meet the principal goal of this dissertation are:

- Finding out ways to measure and improve productivity
- Observing the impact that operating agile has on the productivity of the teams
- Examining the benefits of integrating lean and agile approaches in software development industry

- Collecting best practice suggestions on how to manage teams while aiming to improve their productivity and increase business value.

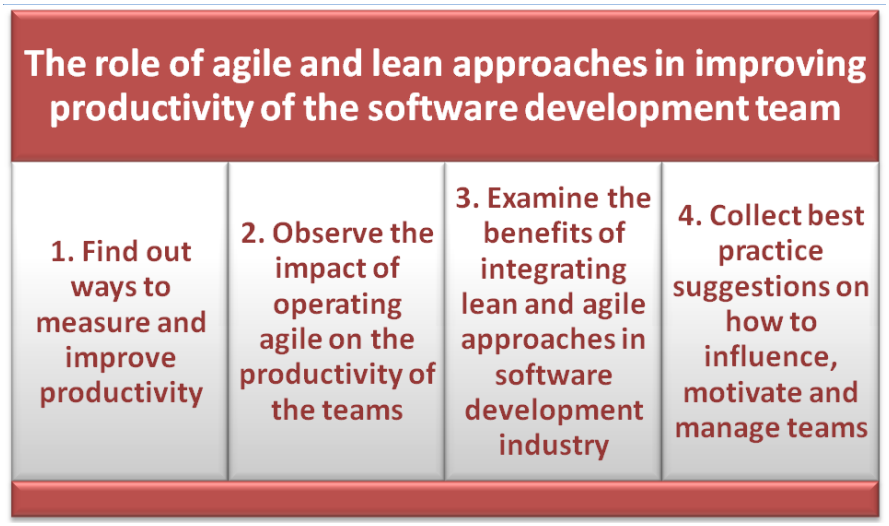

Figure 1.1. Aim and objectives

1.3. Research method

This research is based on the collection of primary data using different tools such as interviews and questionnaires (Hart, 2012). Firstly, the theoretical framework is created and it is organized in four subsections that include productivity, agile, lean and soft management issues. Afterwards, a qualitative methodology is used to conduct the study. The qualitative approach is mainly based on interviewing Scrum masters, representatives of several project-driven organizations in various countries of the world. An online questionnaire that contains questions of the same nature with those of the interviews, is using as a supportive tool for the study,. Afterwards, the results of the study are analyzed and discussed comparing the current findings with those of previous researchers whose work is used as a supporting theoretical framework for this thesis. The study ends in a conclusion section that summarizes the results of the current research. Figure 1.2 displays the research structure of this study.

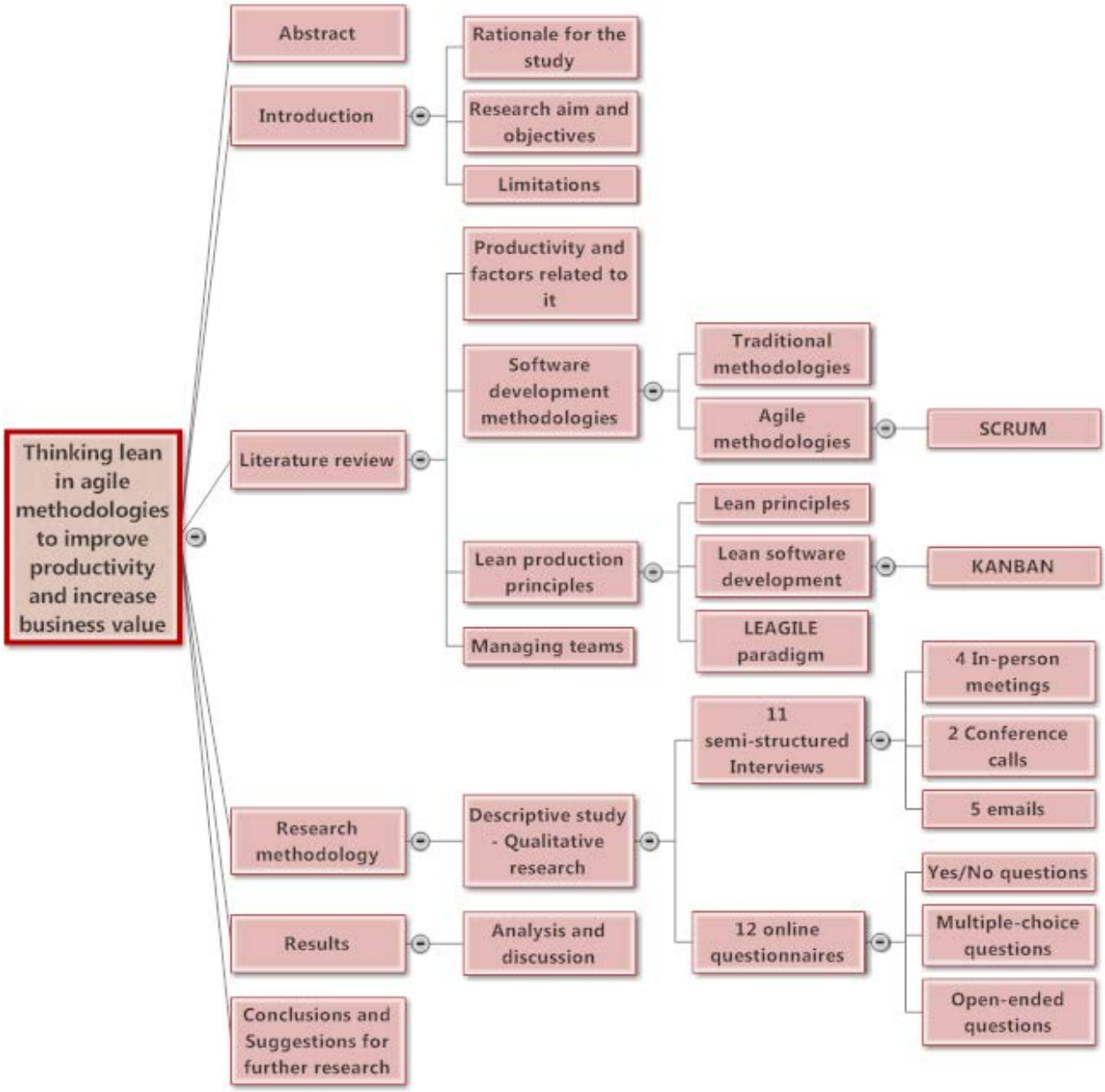

Figure 1.2. Research structure of this study

1.4. Limitations

By conducting research in this Master thesis, the author focuses on those project management practices that have a direct impact on the productivity of the teams and an indirect one on the business value of the organizations that run these software development projects. Agile and lean tools (such as Scrum and Kanban), offer viable solutions to the changing requirements and specifications at every stage of software development projects. A possible limitation according to Wang, *et al*. (2012) comes because of the broad and poor definition that the concepts “agile” and “lean” have. This can perhaps have its impact

on the organizations that aim to combine these approaches for the benefits of their working processes.

The fact that all the interviewees are Scrum masters is another potential limitation. Involving various participants running Scrum projects in different parts of the world makes the study fascinating. However, if the time would be longer, the involvement of more participants would make the findings more accurate and interesting.

2. LITERATURE REVIEW

2.1. Software Development Productivity

2.1.1. Why do software projects fail?

Software development projects have continuously experienced several serious difficulties during their life-cycle. Lindstrom and Jeffries (2004) point out several factors that cause software development projects to fail such as: unclear, wrong or changing requirements; untestable/unmodifiable software code; etc. All these difficulties often cause the software projects to get out of hand in the development cycle (Tsoi, 1999). This can be translated in cost overruns, delays of final product release and customer dissatisfaction (Chiang and Mokerjee, 2004), which cause many projects to be aborted, make them miss the implementation of some minor components, or do not permit them enough debugging to the delivery time (Tsoi, 1999). Therefore, it is important to find the appropriate ways of enhancing the productivity in this field.

In order to assess the right path of improving productivity, it is crucial to set some objective criteria to measures it. “You cannot control what you cannot measure.” state Anselmo and Ledgard (2003, p. 121) in their paper. Several researchers stress the importance of productivity measurement and improvement as a main factor to produce high quality product at a lower cost (Collofello, *et al.*, 1983). On the other side, high quality product at a lower cost can be translated in more value created for the money invested in software projects; seeing the whole picture from a managerial perspective, this is what mostly matters: the monetary aspect of the analysis of productivity (Petersen, 2011).

2.1.2. Productivity definition and measurement

Productivity is often defined as the amount of output per unit of input used (Kitchenham and Mendes, 2004; Petersen, 2010).More specifically software development productivity is defined as the ratio of product size to project effort (Kitchenham and Mendes, 2004). However, there are not standardised productivity benchmarks in software industry. The most common measures of software products, as Anselmo and Ledgard (2003) point out, are functionality, complexity and quality. These measures serve to determine productivity, which on the other side depends on the understandability and independence of the modules

produced (Anselmo and Ledgard, 2003). Duncan (1988) defines productivity as the time and/or cost required to deliver those software products that meet the need of the customers, focusing the attention on the quality of the final output, and on the quantity of the software produced for a given time, with a given cost.

Maxwell and Forselius (2000) support previous researchers' work, defining productivity as output divided by the efforts to produce that output, taking into account that the output measurement should be a function of projects' size, quality and functionality. According to several researchers such as Collofello, *et al.* (1983), Duncan (1988), Maxwell and Forselius (2000), Petersen (2010), some common criteria used to measure productivity are:

- line-of-code per programmer,
- function-point counts,
- number of test case runs,
- pages of documentation per programmer,
- implemented changes ,
- the cost of development and maintenance,
- the predictability of the schedule
- the realization of the pre-set milestones,
- the number and types of project reviews

Historically, several developers have used several approaches to measure the software development productivity (Scacchi, 1995; Ferrão, 2010; Maxwell and Forselius, 2000; Petersen, 2011; Stroustrup, 2012). Among the most common ones are

- **Lines-of-code productivity** – expresses productivity in terms of work units and consists of new and/or modified lines of code as output, and many efforts to develop and maintain the existing lines of code (Scacchi, 1995). The production of quality code is crucial for the software development (Stroustrup, 2012). However, this method has two main drawbacks that complicate the use of this kind of productivity in benchmarking: making explicit the components that should be considered and the variance of the measure within the same programming language or between several ones. (Maxwell and Forselius, 2000)
- **Function points productivity** – productivity is measured based on the output, which consists on a number of function points (Symons, 1988; Ferrão, 2010). It is

often subject to bias as there might always be counting errors, the counting process might be influenced by a subjective judgment, etc. (Petersen, 2011)

According to Chiang and Mokerjee (2004), software, because of being intangible, is almost impossible to measure. Three main aspects of software management should be considered when aiming to improve software development productivity: people, technology and processes (Chiang and Mokerjee, 2004; Lindstrom and Jeffries, 2004).

2.1.3. Software Lifecycle and Development time

According to Duncan (1988) software development project is mainly composed by six major stages: Requirement analysis, Design, Implementation, Testing, Support and maintenance, and Retirement (Duncan, 1988).

Time is an essential factor when aiming to improve software development productivity. In a rapidly developing world, full of potential alternatives, the companies have to struggle to maintain and grow the market share. Blackburn, *et al*., (1996) argue that the customers will not wait for the product delays, therefore time overruns and low productivity tend to fall to the bottom line, as when the competitive market offers substitutes, the delays can be devastating. The study of these scholars make one realize about the importance of time in software development projects and boost the curiosity of the reader to see what value is given to the “time” factor nowadays, and what is the impact of this factor on productivity.

2.1.4. Team size

Various studies and observations have been made in order to find out if the team size affects the software development speed. The faster firms, known also as “tiger teams” in literature, tend to have smaller teams (Blackburn, *et al.*, 1996).

Another argument that Blackburn, *et al.*, (1996) rise, and that is further supported by Chiang and Mokerjee (2004), is about the relation between team size and productivity. According to these scholars, increasing team size decreases the productive output per team member and leads to a longer project duration due to the fact that larger teams require more coordination among their members. to establish and maintain a good communication among them; this somehow leads to a decrease in productivity (Blackburn, *et al*., 1996; Chiang and Mokerjee, 2004).

Some years later, Bosch and Bosch-Sijtsema (2010) claim that the ideal size of the team consist in a number of less than 10 people. Schwaber and Sutherland (2011, p. 6) go on defining the optimal team size as “small enough to remain nimble and large enough to complete significant work”, proposing a development team of three-nine members. Teams compounded by less than three people drecrease interaction and productivity, while those with more than nine members require many efforts too coordinate and make the empirical process difficult to manage (Schwaber and Sutherland, 2011).

2.1.5. Team skills

Special importance on the team skills, with a focus on productivity, has been given by Chiang and Mokerjee (2004). The authors argue that the knowledge and experience of team members is crucial to the development speed as an experienced team requires less effort to coordinate and adapt to the components of the systems during the project. Reflecting on previous research studies, these researchers claim that there is a clear productivity gap between beginners and expert developers (Chiang and Mokerjee, 2004).

2.1.6. Change management

Since software development takes place in a dynamic and volatile environment, and since the projects have to adapt to changing circumstances, the management of those projects must also adopt to changes that are related to several aspects such as technologies, people and requirements. As Tsoi (1999) states, such changes may consist in the adoption to different project life cycles, distributing the development, or moving to component based one. As a result, change management is important in software development industry as it helps the project managers to monitor, control and coordinate their projects (Tsoi, 1999).

2.1.7. Technology and tools

Technological innovations and tools, as Chiang and Mokerjee (2004) point out, affect the program comprehension support that the project environments provide to the team members and the extent to which the team members can combine development and coordination. New tools and facilities boost the software development speed and reduce coordination overheads. (Chiang and Mokerjee, 2004)

2.2. Traditional versus Agile approaches

2.2.1. Traditional software development methodologies

Traditional approaches were designed to be innovative and effective in the context of the existing technologies (Livermore, 2008). Overloaded with documentation and development procedures, these methodologies are suitable for static business requirements, as changes in the customer requirements require prior change to the existing software products and documentation.

- ***Waterfall model*** – considers the project as a set of subsequent phases that must be articulated, documented, executed and signed off, the most important of which is the one in which the team members have to deal with the customer requirements. This methodology was initially seen as mandatary to be applied in the case of the large software development projects. (Northover, *et al.,* 2007)
- ***B – model*** – an extention of waterfall model, but with an extended lifecycle to ensure that the development stages would constantly include constant improvement of software (Ruparelia, 2010).
- ***Iterative Incremental Development*** (later evolved into ***Rational Unified Process***) – is a shift from the traditional waterfall model which relies on object-oriented programming. (Northover, *et al.,* 2007)
- ***Spiral model –*** attempts to address the difficulties of waterfall model by analyzing and classifying risk (Ruparelia, 2010).

The typical feature of the traditional methods of software development (i.e. they promoted a strong conformance to plan, encouradged a strong division of roles among the team members, relied heavily on the repeatability of the process, etc.) led to the embracement of "agility", a more humanistic and collaborative philosophy of developing software (Maurer and Melnik, 2006).

2.2.2. Agile methodologies

Firstly introduced by the end of '90s as a reaction of the challenges that came with the Internet era and with the usage of "heavyweight" methods (Northover, *et al*., 2007), agile approaches are nowadays wellknown throughout the software development industry as "mainstream" software engineering (Maurer and Melnik, 2006; Mishra and Mishra, 2011). These methods revolutionized the working processes in software development industry

(Figure 2.1), as they removed barriers and facilitated the implementation of the changing requirements by quickly responding to them; established a close and ongoing collaboration between the development team and the customer (Germain and Robillard, 2005; Livermore, 2008); met short product cycles, as the today businesses demands (Livermore, 2008). Since the clients' requirements cannot be anticipated from the very beginning of a project, some flexibility should be built within the development activities in order to meet the foundamental criteria of the iron triangle: time, cost and quality (Germain and Robillard, 2005).

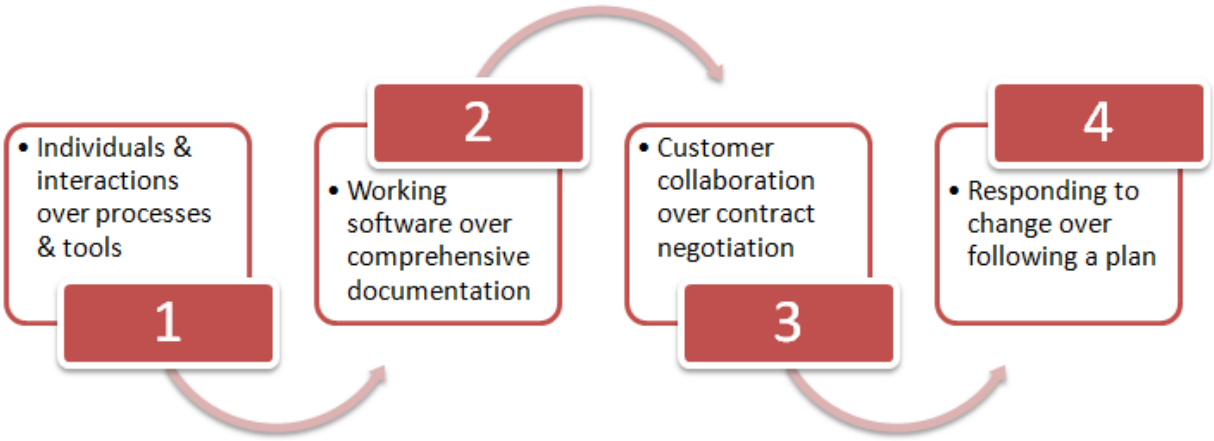

Figure 2.1. Agile methodologies over Traditional ones according to (Beck, *et al.*, 2001)

The importance of agile methodologies has been increasingly wellspread over the last years as they provide lower cost and better quality, productivity and customer satisfaction (Mishra and Mishra, 2011) Indeed, these methodologies have proved to be more efficient than the traditional ones (especially in projects of modest to moderate size (Bosch and Bosch-Sijtsema, 2011)), as they offer more flexibility and agility that helps to produce higher quality software in a short period of time (Mishra and Mishra, 2011). Agile approaches assume active customer involvement as crucial to ensure the successful implementation of the right functionalities. (Bosch and Bosch-Sijtsema, 2011).

Common features of agile methodologies are prototyping, minimal documentation and iterative development (Livermore, 2008). They all rely on the transparency and honesty of the working code which shows developers what they really have in fron of them, and on the effectiveness of people collaborating with goodwill (Highsmith and Cockburn, 2001).

Agile Manifesto approach consists on a number of methodologies such as Scrum, the Cristal methods, Feature-Driven Development, Adaptive Software Development, Dynamic

System Development, Extreme Programming (XP), Extreme Project Management, Lean Development (Beck, *et al.*, 2001; Highsmith and Cockburn, 2001; Lindstrom and Jeffries, 2004, Germain and Robillard, 2005; Livermore, 2008; Bosch and Bosch-Sijtsema, 2010; Bosch and Bosch-Sijtsema, 2011; Dingsøyr, *et al.*, 2012; Wang, *et al.*, 2012), and so far (Figure 2.2).

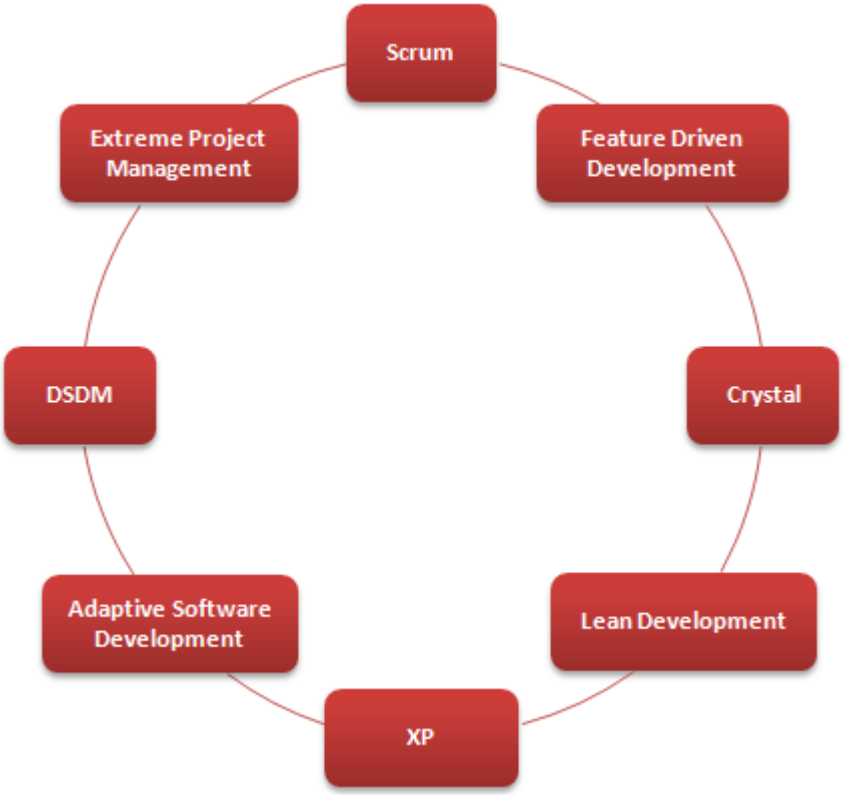

Figure 2.2. Agile methods according to Lindstrom and Jeffries (2004)

- ***Crystal Methods*** – a toolkit used to align the appropriate methodologies to the individual projects, alwayes assuming that people more than technology and processes are crucial to the software development projects (Livermore, 2008). Highsmith (2002) points out three main factors impact the amount of the methodology elements that should be used in a software development project:
    1. The communication among the team members, which is affected by the location of the individuals, their personality features and the office layout.
    2. The negative and threating implications that might come from the presence of undiscovered software deffects after the software release.
    3. The presence of the organizational priorities that might impact the development process. (Highsmith, 2002; cited in Livermore, 2008).

- ***Feature Driven Development*** – a paradigm that, as Figure 2.3 shows, has as a main principle the composition of products from individual features (Kircher and Hofman, 2012). Livermore (2008) describes it as a procedure that starts with the development of a whole picture of the desired application, goes on with the determination of the basic features that this application should contain, then with the development of an implementation plan which prioritises the list of features, and ends with the executation of this plan. Each development iteration produces deliverables for the customer, while causing a repriotization of the features in the remaining list in order to keep the development team working on those features that the customer retains as the most important and business value adding ones. (Livermore, 2008)

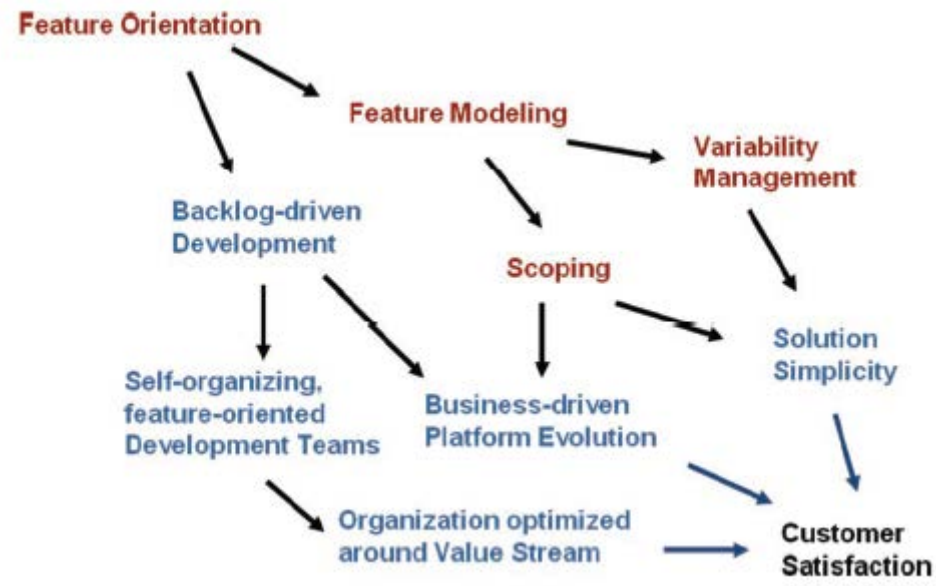

Figure 2.3. Feature-orientation as foundation for Agile (Kircher and Hofman, 2012, p. 217)

- ***The Whitewater Iterative System Development with Object Modules (WISDOM)*** – an agile method suitable for the small companies that utilizes an iterative process of refining a prototype. (Livermore, 2008)

- ***Extreme Programming (XP)*** – an approach based on simplicity, communication, feedback and couradge (Lindstrom and Jeffries, 2004) that focuses mainly on the implementation of software (Mishra and Mishra, 2011) and makes it possible to cope with change throughout the software development lifecycle (Northover, *et al.*, 2007). This approach is characterized by several practices such as: 1-2 weeks iterations, pair programming, shared code, continuous code integration, reflection,

face-to-face interaction, small releases, designing tests before developing code, refactoring, and stand up meetings (Northover, *et al.*, 2007; Livermore, 2008; Mishra and Mishra, 2011). The customer (the person who sits and works on-site with the development team, representing the business requirements and values, and the acceptance criteria of the project) is central to the methodology (Lindstrom and Jeffries, 2004). It is important for both pairs (developers and customers) to possess good interpersonal skills and trust for the effectivity of the working process (Dyba and Dingsoyr, 2009).

> The essence [of XP] truly is simple. Be together with your customer and fellow programmers, and talk to each other. Use simple design and programming practices, and simple methods of planning, tracking, and reporting. Test your program and your practices, using feedback to steer the project. Working together this way gives the team courage. (Jeffries, 2001 p.172, cited in Lindstrom and Jeffries, 2004, p. 45)

The main benefit of XP approach is that it helps towards an effective risk management process from the early phases of a project and it also helps towards the system growth in an incremental way. (Cordeiro, *et al.*, 2008)

- ***Scrum*** – a methodology that aims to improve the communication between team members by focusing more on the management aspect of the software projects than on their technical details (Mishra and Mishra, 2011), assuming that variables related to people and technologies are likely to change during the process (Cordeiro, *et al*., 2008). This approach uses time-boxed events, making sure that each of them has a maximum duration in order to ensure that the right time is invested in planning, eliminating waste during this process (Schwaber and Sutherland, 2011). Scrum is implemented by breaking a project into and one to four week iterations called “sprints” (Livermore, 2008; Cordeiro, *et al.*, 2008), which are Scrum events that represent an opportunity to inspect and adapt change, enabling critical transparency and inspection (Schwaber and Sutherland, 2011).

2.2.2.1. A closer look into Scrum

A. The Scrum meetings

Scrum teams organize several meetings to plan and reflect on their work (Figure 2.4). According to Cordeiro, *et al*. (2008), two main meetings are hold to plan the sprints:

- In the first meeting, the product owner refines and reprioritizes the *product backlog*, which contains a list of features, defects, improvements, stakeholders[1] and targets to be achieved.
- In the second meeting the Scrum team creates *the sprint backlog* that contains the detailed tasks that have to be accomplished during the current iteration.

Additional meetings organized by Scrum teams are "sprint reviews" and "sprint retrospectives". By the end of each iteration, the Scrum team represents the results achieved during the iteration (the finished sprint that they have been working on) to the product owner, customers and other stakeholders (Cordeiro, *et al.,* 2008), in a meeting called "*Sprint review*", the results of which function as a basis for the product owner's inputs to the following *Sprint Planning* meeting (Ovesen, 2012). "*Sprint Retrospective*" meetings, part of the "inspect and adapt" Scrum phylosophy for continuous improvement, follow immediately after the "*Sprint Review*" meetings and serve to identify those part of Scrum that need to be modified or changed in order to improve the collaboration between team members, the conditions of the working environment and the efficiency of the working processes (Ovesen, 2012).

Scrum daily stand-up meetings, the most common and important ones with a duration of 15-30 minutes, are supposed to be short, procise and concise and to focus only on the sprint tasks (Dinakar, 2009). These meetings serve for team members to raise question and discuss several issues with the other colleagues (Cordeiro, *et al.*, 2008; Ovesen, 2012) Each developer talks about the progress of his/her work and what is left to be done, as well as the bottlenecks/dependences that may cause difficulties during the work (Dinakar, 2009).

[1] In software development industry the term "stakeholders" is limited to developers, the customers, end-users and project sponsors (Power, 2010)

	1 SPRINT PLANNING	2 DAILY SCRUM	3 SPRINT REVIEW	4 RETROSPECTIVE
CHARACTERISTICS	Forward-looking	Corrective	Controlling	Retrospective
MAIN FOCUS	Breakdown and estimation	Burn-down of tasks	The Product	The Process
PURPOSE	Planning	Team communication	Showing work done	Improving the process
PARTICIPANTS	Scrum Team	Dev. Team & Scrum Master	Dev. Team & stakeholders	Dev. Team & Scrum Master
INPUT	Product Backlog Items	Development status	Work done in the Sprint	Process-related problems
OUTPUT	Sprint Backlog	Team synchronisation	Input to Product Backlog	Plan for process improvement

Figure 2.4. Scrum meetings

B. Cancelling a sprint

A sprint cancellation, which according to Schwaber and Sutherland (2011) is done only by the product owner, should occur before the time-box is over, and only if the Sprint goal becomes obsolete. However, sprint cancellations should be very uncommon, as they consume resources and might cause stress for the team (Schwaber and Sutherland, 2011).

C. The Scrum team

The Scrum team is compounded by the Scrum master, the product owner and the development team (Schwaber and Sutherland, 2011).

1) *The Scrum master* – an individual (similar to the project manager in the traditional project management practices) that acts as a "servant-leader" to facilitate the work of the development team, making sure that they have understood the Scrum theory, practices and rules (Schwaber and Southerland, 2011).
2) *The product owner* – an individual (similar to the project sponsor in the traditional project management practices) that is responsible for the Product Backlog, for maximizing the value of the product and for the work of the development team (Schwaber and Sutherland, 2011; Ovesen 2012).
3) *The development team* – Unlike the traditional project management teams where the project manager takes the main responsibility, in a Scrum team the responsibility is fully and equally shared among the members of the development team (Ovesen, 2012). Within the Scrum framework, the development teams are self-managing (Lindstrom and Jeffries, 2004, Dinakar, 2009, Schwaber and Sutherland, 2011), cross-functional (Ovesen, 2012), co-located (Bosch and Bosch-Sijtsema, 2010), and continuosly improve the development processes by providing self-feedback (Dinakar, 2009). These

features represent optimal conditions to an effective and efficient development process (Schwaber and Sutherland, 2011; Ovesen, 2012).

Self-directed teams are effective because decisions are made by the team-members who are close to the details and have an overall understanding of the project goals; they are supported by the managers who are part of the communication loop (Lindstrom and Jeffries, 2004). The team are empowered to decide the priority of the tasks that they wish to deliver (Highsmith, 2002) and to adjust the development process over time (Talby and Dubinsky, 2009). Self-organized teams are proved to be successful as they boost the motivation of the people involved in them (Bosch and Bosch-Sijtsema, 2010).

As for the cross-functionality of these teams, Ovesen (2012) reflecting on Keller's (2001) research, states that it has its advantages which consist on better technical, schedule and budget performace; but it also has its drawbacks that mainly consist on increased job stress and lack of focus. Increased job stress because the developers have to collaborate closely with individuals of other educational, functional and cultural backgrounds, often under speed-to-market pressure. Lack of focus becase multitasking among different projects may lead to a decrease of the efficiency of a developer (Ovesen, 2012).

Other important factors for the productivity of the development team are the size of the team and the proximity of the team members. As discussed earlier, an optimal size is between three and nine members per team, without including the Scrum master and the product developer, unless the last ones are executing the list of tasks stated in the Product Backlog (Schwaber and Sutherland, 2011). As for the co-location, the whole team should work together and have face to face contact (Bosch and Bosch-Sijtsema, 2010), in an "open workspace", the walls of which are used to display information[2] about the project in order for the progress to be seen by all the participants and stakeholders, which will be able to judge the progress of work and the productivity of the team (Lindstrom & Jeffries, 2004). Dispersed team could be a challenge as Scrum works better for the local software development teams as it promotes the close collaboration, communication, support, coordination,

[2] Information about acceptance tests, project status, etc. is displayed in whiteboards and charts of metrics (Lindstrom and Jeffries, 2004).[3] Value stream is a flow of activities from forecasting and planning to the complete accomplishment of the production process. (Narang, 2008)

development effort and team cohesion of the colocated teams (Ovesen, 2012). Distributed teams compensate face-to-face meetings and try to buid synchronous communication through technological means such as phone/video-conferencing (Bosch and Bosch-Sijtsema, 2010).

D. Scrum artefacts:

1. *Product backlog* – a dynamic document hold by the product owner that consists in a list of functional and nonfunctional requirements of the product (Dinakar, 2009) that explain what should be done in order to improve the product in the upcoming releases (Schwaber and Sutherland, 2011).
2. *Sprint backlog* – a dynamic document that evolves throughout the Sprint and represents the list of task (with the respective time needed) to be accomplished by the development team in order to meet the Sprint goal.
3. *Scrum board – he*lps the team to set and maintain a proper communication among the developers, by using post-it notes, or writing charts, graphs and other tasks-to-be-accomplished in a whiteboard and making it visible to all the team members.
4. *Burned down chart* – used in relation to the Product/Sprint Backlog to represent a raport of the work left versus the time left. (Ovesen, 2012)
5. *Product increment* – represents a number of the Product Backlog items that have to be sent to the Sprint Backlog for development, and removed from the first list afterwards. (Ovesen, 2012)

Scrum methodology has shown to be successful when being used with both small and large projects which on the other side, can be broken into subprojects under the control of the respective assigned Scrum teams which are coordinated through Scrum techniques (Livermore, 2008). Yet, the skills and the motivation of people involved in the project is crucial to the productivity and the quality of the final result (Cordeiro, *et al.*, 2008).

To sum up, Scrum is considered as the most appropriate agile approach to enhance the work of the software development teams. It helps the developers to organize and keep on track their work. With the help of its artefacts, Scrum enhances the communication between team members by bringing them together in the same working environment. The autonomy that developers are given increases their

creativity and leads to a raise in productivity. Even though this methodology might be associated with challenges, the introduction of the "sprint" concept improves evidently the working processes helping teams to deliver in time according to the customer expectations and avoiding the risk of an outdated product.

2.2.2.2. The implementation of Agile methodologies

Migrating to agile methodologies involves several aspects of an organization including people, technology and processes (Nerur, *et al.*, 2005). Several factors, displayed in Figure 2.5, have a considerable impact on the implementation of an agile methodology (Livermore, 2008). However, the positive effects of implementing agile are tangible for the organizations that integrate agile principles in their developing processes, even though they often come with several challenges.

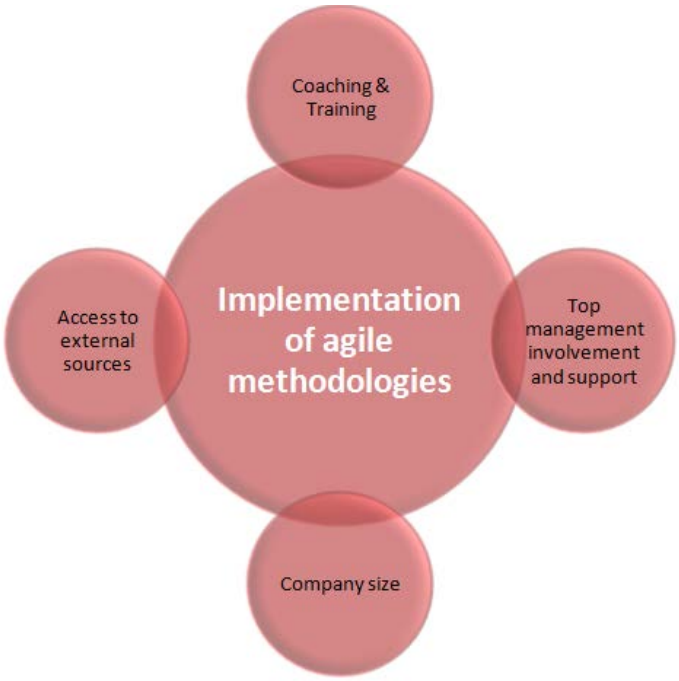

Figure 2.5. Factors that affect the implementation of an agile methodology (Livermore, 2008)

A. Benefits

Agile approaches have proved to increase customer satisfaction, enhance product quality and improve the organizational morale (Mishra and Mishra, 2011). They accelerate the production of functionalities that are important to the customer at present (Fogelström, *et al.*, 2010). Two additional values that make agile methodologies superior to the traditional ones, as Ambler (2009) asserts, are the greater visibility of the stakeholders into the project and the setlement of decision gates in form of milestones in order to reduce risk. Indeed,

Rico (2011) agree on the capability of agile methodologies to handle an effective risk-management process. He claims that their disciplined, flexible and lightweight structure helps to an effective decision-making process; promotes collaboration between developers and developer-customer; makes these methodologies adaptable to continuous change, and focus on bottom line business results for the marketplace. All these positive sides result in lower costs and fast-time-to-market. The frequent involvement of the customer during the developing processes, the early testings and the small releases increase customer satisfaction and trust, and makes them loyal to the business. The abovementioned elements cause companies to achieve a higher return on investement by improving the number of sold products, the revenues achieved from the sales and as a result the overall profit. The close collaboration with the customer makes the work easier for the developing team improving their morale, sustainability and performance. After having stated a numerous number of benefits that the implementation of agile methodologies brings to organizations, Rico (2011) concludes his study pointing out the two main drivers that lead all the others when it is to the integration of agile principles: increased productivity that comes from their streamlined nature, and increased quality that comes from their uncompromising discipline. (Rico, 2011)

B. Challenges

Despite its benefits, the applicability of the agile methods is often considered challenging, due to several factors. One of the most important one is knowledge sharing. Agile methods rely heavily on the informal communication and collaboration among the developers and customers to access and share tacit (more than explicit) knowledge between them (Chau, *et al.*, 2003; Nerur, *et al.*, 2005). This might be a problem in large projects due to the large number of stakeholders, and due to the large amount of information that flows among the members (Mishra and Mishra, 2011). If the traditional models facilitate knowledge sharing through documentation (Chau, *et al.*, 2003), where it is well specified ‘what’, ‘how’ and ‘when’ a certain task should be done, the lean and mean documentation that agile approaches provide promote mainly the self – documenting designs and self-describing code requiring a closer communication and collaboration within the team (Nerur, *et al.*, 2005).

A significant factor to be considered when commiting to an effective implementation is the allocation of the necessary resources to help make the cultural change to agile (Livermore,

2008). The culture-sensitive elements that include dialog messages, error messages, and menu names are stored in a message file that differs throughout markets/cultures (Yeo, 2001).

It is important to have the commitment of the people involved in these projects when aiming to adapt new methodologies. After all, the results of the overall process depends on their work. Lindstrom and Jeffries (2004) found out that the developers who did not embrace the new methodologies appeared to be undisciplined and indifferent to quality, no matter if they managed to deliver the software that the customer had asked for in the beginning. Other scholars see agile approaches as tactical ones, and therefore the major changes required to turn the working processes into agile ones, must be initiated from the top of the organization (Wang, *et al.*, 2012). Consequently, the support of top management, in terms of providing necessary technical financial support together with employee empowerment, is crucial for the effective implementation of agile principles in the organizational strategy (Gunasekaran, 1999).

Another common challenge, noticeable mainly in large-scaled complex projects, is the fact that agile methods involve the customers directly in acquiring requirements and domain knowledge (Chau, *et al.*, 2003; Nerur, *et al.*, 2005). The complexity of the application domain, which is is beyond the expertise of both customers and developers, leads to a risk of running out of time and resources (Mishra and Mishra, 2011).

When talking about large-scaled projects, Bosch and Bosch-Sijtsema (2010) argue that the project team can face additional challenges when trying to adapt agile. This is first, due to the fact that large component teams require extra efforts to coordinate with the other teams working on the same project. Secondly, due to the fact that they focus on their specific component of the product, developers often shift focus or miss information about the customer's picture of overall product. This inward focus causes stress among the team members and the product managers who have to face the customers at the same time. Thirdly, in order to test a product, all the functionalities should be completed and coordinated, but misunderstandings among the project team, or incoherences in sequencing can cause delays and unpredictability (Bosch and Bosch-Sijtsema, 2011).

Benefits	Challenges
(1) requirements are more precise due to reduced scope and thus easier to estimate, (2) direct communication in teams reduces the need for documentation, (3) early feedback due to frequent deliveries, (4) rework reduction, (5) testing resources are used more efficiently (6) higher transparency of who is responsible for what creates incentives to deliver higher quality (7) low requirements volatility in projects (8) reduction of waste (discarded requirements) in the requirements engineering process.	(1) challenges in regard to realize continuous testing, (2) increased maintenance effort with an increase in the number of releases, (3) management overhead due to the need for coordination between teams, (4) detailed dependencies are not discovered on a detailed level due to lack of focus on design, (5) long requirements engineering duration, due to complex decision processes in requirements engineering, (6) requirements priority lists are hard to create and maintain, (7) waiting times in the process, specifically in design waiting for requirements, (8) reduction of test coverage due to shortage of projects and lack of independent testing, (9) increased configuration management effort.

Table 2.1. Benefits and Challenges of implementing Agile (Mishra and Mishra, 2011)

Considering the benefits and challenges of operating agile, several questions come to the attention of this study (Figure 2.6).

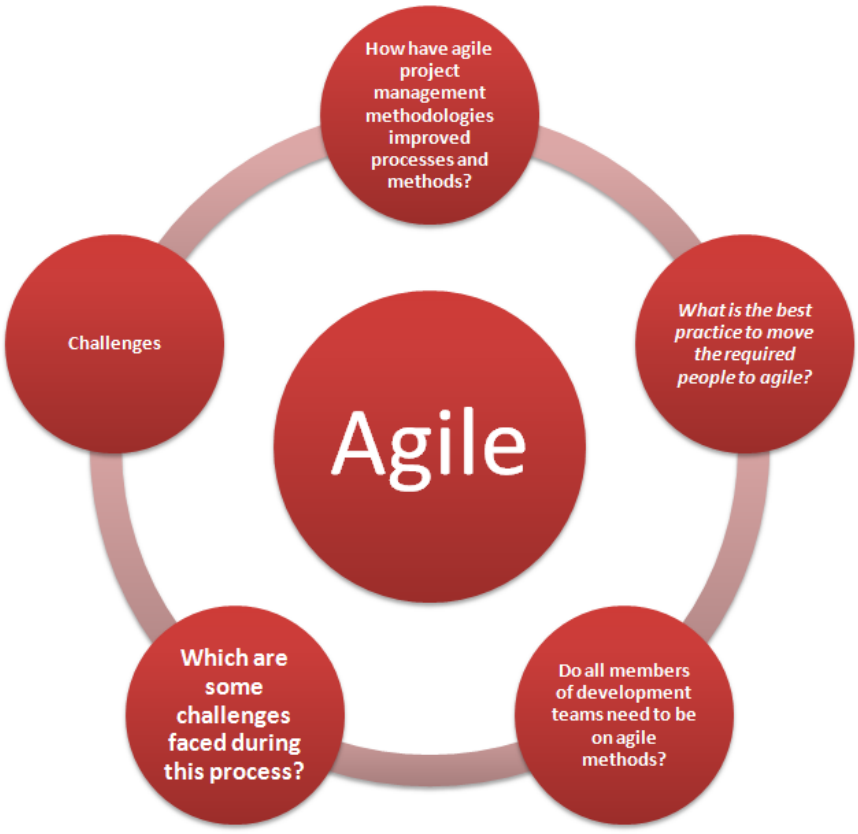

Figure 2.6. Question drawn from theory

To sum up, traditional approaches are favorable only in those projects where the customers' requirements are clear from the beginning. In a volatile environment, agile methodologies benefit the working processes by ensuring close collaboration with the

customer, guaranteeing a fast and effective delivery and avoiding the risk of delivering late and failing to meet the user expectations. As any other methodologies, they come with several changes, which are mainly related to the fact that embedding these approaches changes the whole organizational mindset. However, if handled properly, agile methodologies benefit the business and enhance the motivation and the productivity of the teams.

2.3. Embedding Lean

2.3.1. The lean philosophy

Agile consists on using market knowledge to exploit profitable opportunities in a changing marketplace (Lu, *et al.*, 2011)., while lean production consists on developing a value stream[3] to reduce costs and improve quality, by eliminating waste (Mateosian, 2003, Conti, *et al.*, 2006; Sim and Rogers, 2009; Lu, *et al.*, 2011). In lean philosophy, waste (Figure 2.6) is defined as an activity that takes time but does not add customer value (Naylor, *et al*., 1999; Liker, 2004; Conti, *et al*., 2006; Petersen and Wohlin, 2008), affecting business performance instead and threatening its prosperity, unless it is continuously and eliminated (Emiliani, 1998).

[3] Value stream is a flow of activities from forecasting and planning to the complete accomplishment of the production process. (Narang, 2008)

Figure 2.7. Non-value-adding waste in business according to Liker (2004)

Lean production was firstly introduced by Toyota, therefore Toyota Production System (TPS) is the basis of lean principles. Citing Womack's and Daniel's (1996) work, Liker (2004, p. 7) defines lean manufacturing as a five-step process (Figure 2.7) that consists on:

1. Defining customer value
2. Defining the value stream[4]
3. Making the product "flow" from value adding processes without interruptions
4. Creating a "pull" system that cascade back from the customer demand
5. Building a learning culture that strives for excellence and improvement

[4] The definition of value stream depends mostly on a customer and cost perspective rather than on the organizational point of view (Krishnamurthy, *et al*., 2007).

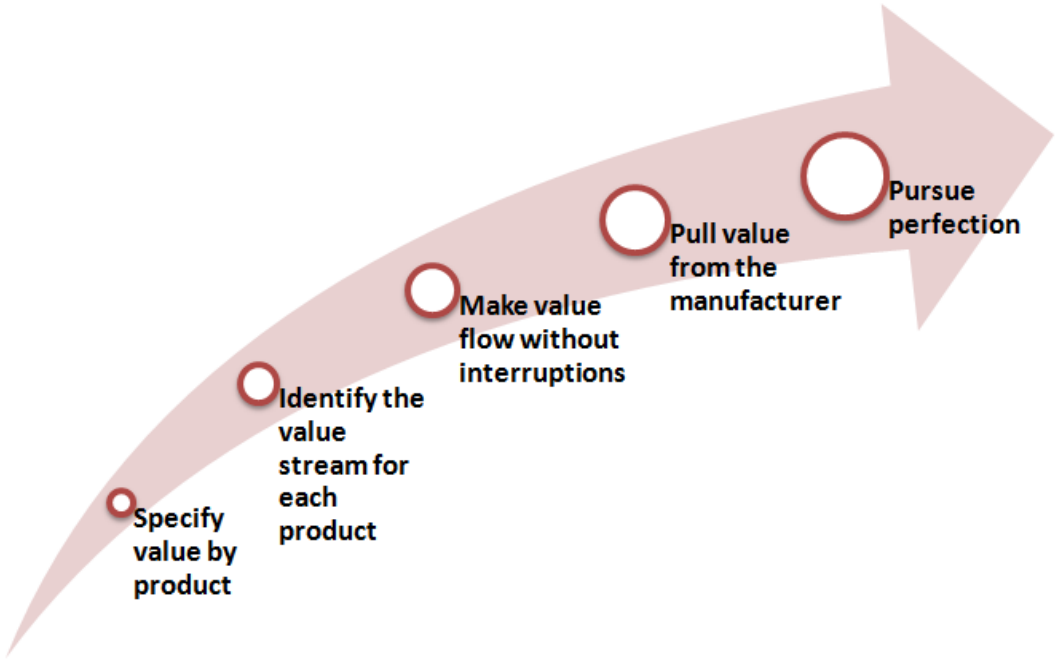

Figure 2.8. Basic principles behind lean thinking (Womack, 1996)

Liker (2004) and Liker and Meier (2006) go further creating a "4-P" model that correlates the high-level principles of Toyota with four categories of Philosophy, Process, People/Partners and Problem Solving, with the commitment of top leadership as an indispensable factor throughout the whole process (Figure 2.8)

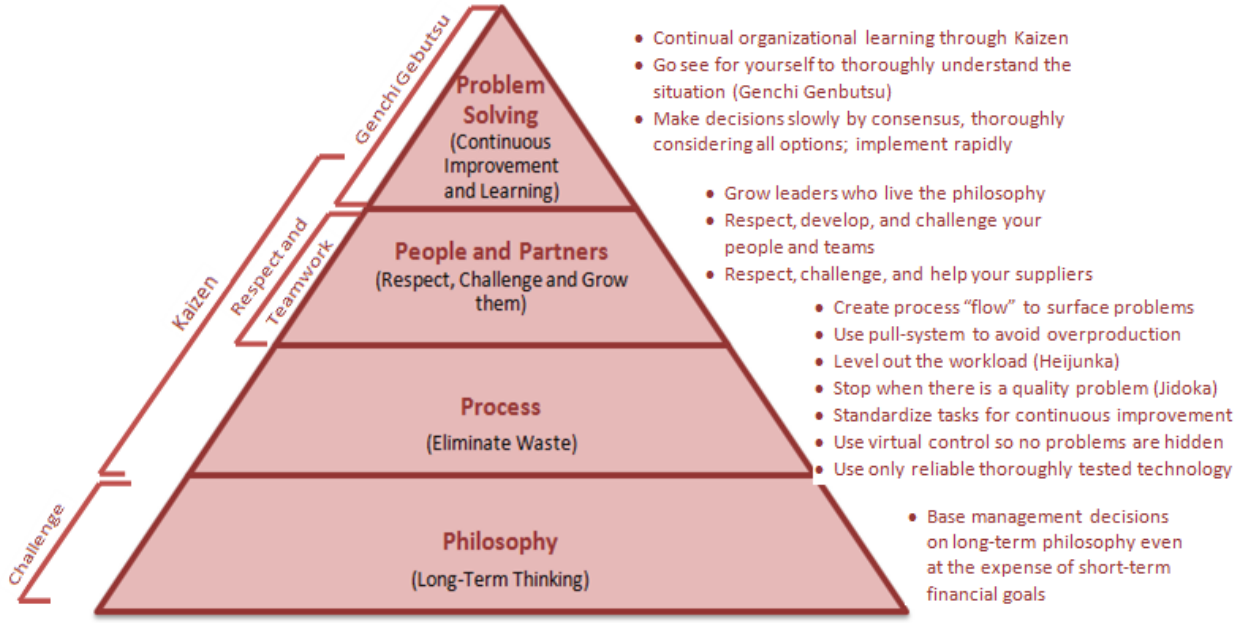

Figure 2.9. "4-P" Model of the Toyota Way according to Liker (2004, p. 6)

The principles of TPS (Figure 2.9) focus more on achieving high levels of customer satisfaction aiming to improve continuously the quality, the efficiency and the flexibility of

the operations (Pil and Fujimoto, 2007), adding value to the working process by aiming to find out in every process what the customer wants from it (Liker, 2004). Although Liker (2004) claims that in TPS it's the people that bring the system to life by closely working together in order to solve issues and make the company grow, Pil and Fujimoto (2007) point out that this superior organizational performance was often at the expense of employees' wellbeing, as it often might increase their stress (Stewart, *et al.*, 2009). According to these researchers, the employers did not put all the efforts on enhancing worker satisfaction by humanizing the working processes and creating more attractive jobs. A couple of years later, Riezebos and Klingenberg (2009) address this issue by noticing that human management and scientific management, instead of contracting each other, could be combined to produce better choices in design, operation and control of production processes.

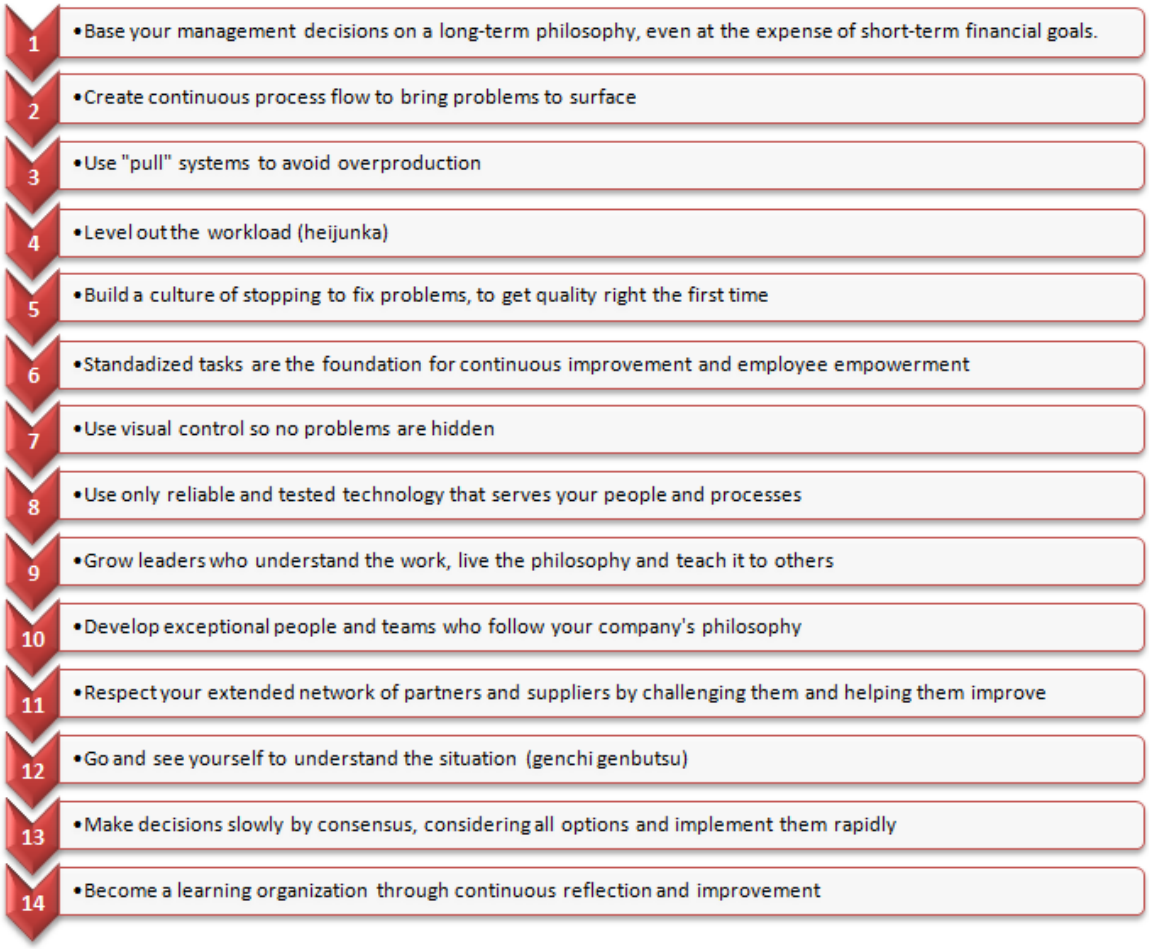

Figure 2.10. TPS principles according to Liker (2004)

2.3.2. Implementing Lean Production principles

The effective implementation of lean principles involves significant functional and cultural changes in a company (Narang, 2008), as well as new approaches to working processes and to serving customers, and a large amount of training and coaching for employees of different hierarchical levels (Sim and Rogers, 2009). During lean implementation, several tools such as pull production (Kanban[5]), reduction, lean layout etc., come into play (Narang, 2008).

Ballé and Ballé (2005) explain the best practices to implement lean principles and suggest that in order for these practices to be effective and to achieve the wished results, managers should work on tacit approaches to create knowledge. Some of the possible ways to do so are the development of specific expertise, the exchange of information among functional specialists (also known as "pull communication" system), the establishment of close collaboration with the suppliers in order to achieve and maintain innovation, etc. (Ballé and Ballé, 2005). Another advisable practice when aiming to incorporate lean is to start from the very last step of the production process as it might be cost-effective and easy to be integrated in the customers' lean strategy (Narang, 2008).

2.3.3. Lean tools and methods

Value stream mapping, is perhaps the main tool of lean production systems. It aims to facilitate and support the implementation of lean philosophy (Lu, *et al.*, 2011) by reducing the non-value adding[6] activities from the ongoing process, identifying all the potential areas that need to be improved (Narang, 2008). Other methods that have been evaluated by various researchers, are summarized and classified in Peter's and Lanza's (2011) work (Figure 2.10).

[5] Kanban, (The word comes form Japanese and means "card/ticket/sign") is a tool for managing the flow in a "pull" system (Liker, 2004). It is an authorization for a process to produce a component based on the consumption of the component by a subsequent process (Lu, *et al*., 2011, p. 30)

[6] Non-value adding activities consume human and monetary resources and extend the throughput time without adding any value to the business. (Narang, 2008)

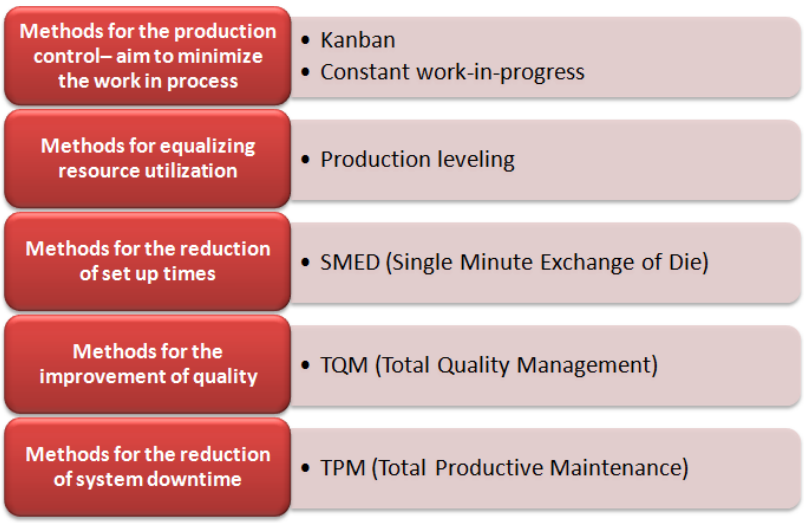

Figure 2.11. Lean methods (Peter and Lanza, 2011)

However, lean production is more than tools and techniques; it is a culture and a philosophy for the whole organization (Narang, 2008), that has as objective to maximize full work, in other words having as many full-working-capacity workers as possible (Shewchuka, 2008). It enables the team to identify problems quickly and fix them permanently (Womack, 1996). It is important for the team to have the support of the team leader when pursuing continuous improvement activities, to be trained in problem solving methods and to undertake cross-trainings in order to carry out cell operations, to have a comfortable workplace and clear standardized working practices (Saurin, *et al.*, 2011).
To sum up, lean production improved significantly the traditional rigid ways of production which used to ignore efficiency and profit of the working processes (Dong, 1995). It helps companies to be highly competitive in the global economy (Dean and Bowen, 1994), offering the best customer service performance (Goldsby, *et al.*, 2006). This is due to the fact that companies have integrated lean principles in their working practices as a way to improve customer satisfaction, productivity (Narang, 2008), quality and delivery performance; create flexibility; reduce cost (Peter and Lanza, 2011) and add to the empowerment and effectiveness of the team members (Braun, 2005).

Yet, this philosophy might have its own limitations. Banomyong, *et al.* (2008) look into previous research and find out that among the main drawbacks of lean production systems are their inability to deal with turbulent and inconsistent change and to cope with the volatile market conditions, as well as their dependency on a stable environment which doesn't allow much flexibility when aiming to pursue perfection.

2.3.4. Lean software development

Lean production principles can be interpreted and adapted to suit different applications and business domains (Fogelström, *et al.*, 2010), therefore they can be transferred from the manufacturing industry to a more abstract and intangible one, which has a totally different culture: software ingineering (Middleton and Joyce, 2012). Lean principles applied in software development, named by various scholars as lean software development, are the same as the agile ones regarding the soft aspect of management (people and leadership), the focus on quality, frequent and fast delivery of value to the customers, etc. (Petersen and Wohlin, 2011). However, lean differs from agile as the lean software development focuses on the end-to-end perspective of value flow through development (Petersen and Wohlin, 2011) and besides, it cannot enable the organization to meet rapidly the changeable customer needs (Agarwal, *et al.*, 2006).

In their study about measurin the flow in lean software development, Petersen and Wohlin (2011) introduce and analyze the tools for eliminating waste and improving the value flow: the value stream mapping, inventory (work in progress management) and pull systems (Kanban). The type of waste that lean principles aim to eliminate in the domain of software development are waiting, extra features and processes, task switching, uncompleted work, movement, defects and unused employee creativity (Poppendieck and Poppendieck, 2003; Hibbs *et al.*, 2009, cited in Wang, *et al.*, 2012).

It has been observerd by several scholars that some lean concepts when applied appropriately to software engineering lead to a number of benefits. One of them is the quick development of quality software with a low cost (Poppendieck, 2007). Middleton and Joyce (2012) bring to attention that applying the lean principles to the software development industry promises other huge benefits that would be measured in terms of a decrease in lead time, variability and error rates, while reducing the work in progress and aiming for continuous improvement. These authors go further claiming that a decrease in lead time translates in shorter development cycles and as a result the customer can quickly receive the desired value, reducing noticeably the market risk, while continuous improvement keeps the team focused to work on a daily basis increasing the predictability of the expected deliverables. Additional benefits that lean development framework were the reduction of the staff turnover, the development of the team skills and the rewritement

of some parts of the legacy code. All the aforementioned positive factors lead to an increase of business value (Middleton and Joyce, 2012).

Other scholars view lean development as a risk management strategy that makes organizations more agile, flexible and willing to accept change (Ambler, 2009). It also promotes team work among individuals who are empowered to make decisions appropriate to their level (Poppendieck and Cusumano, 2012).

Except from "Kanban", not much has been written about how lean concepts can be applied in software industry. A Kanban system is a control system for just-in-time delivery that aims to make work visible, uses the full capacity of the workers to minimize the work in progress[7] by pulling work items from process queues with work limits (Nord, *et al.,* 2012), and helps the work to flow (Wang, *et al.,* 2012). In software, it has more or less the same function, but more specifically it displays information about the status of the process and the potential problems associated with it (Middleton and Joyce, 2012). Poppendieck and Cusumano (2012) explain how a Kanban system works:

> In a Kanban system, the value stream is mapped on a chart (preferably a physical chart on a wall) with columns for each step in the value stream. Kanban cards (tokens for a piece of work) are placed in the column, representing the current state of the work. When work meets specified policies for being complete, the Kanban token moves to the next column, and over time, the card moves from left to right across the board—you visually see work flow through the system. The key to a Kanban system is that work in any column (representing a step in the value stream) is limited. This means that within any value-adding activity, there's a limited amount of work; moreover, the entire system contains a limited amount of work. (Poppendieck and Cusumano, 2012, p. 31)

2.3.5. Leagility paradigm

After having exploited the benefits of agile and lean philosophies, several researchers agree that a combination of these methodologies seems to be potential for huge benefits, since the advantages of both leanness and agility are combined (Cawley, *et al.,* 2010). This would lead to faster responses to market demand using less work force, material and machines (Vinodh and Aravindraj, 2013). In fact, lean production principles have been highly influential in the evolution of agile software development (Greer and Hamon, 2011).

[7] WIP is considered as a form of waste from a lean perspective (Wang, *et al.,* 2012)

For many scholars, such as Mason-Jones, *et al.*, (2000a), Mason-Jones, *et al.*, (2000b), Agarwal, *et al*. (2006), Krishnamurthy, *et al.* (2007), Banomyong, *et al.* (2008); Soni and Kodali (2012); Wang, *et al*, (2012), etc., the starting point and at the same time the motivation to look deeper into the "leagilility" paradigm has been Naylor's *et al.* (1999) work. These researches point out that agility will eliminate waste but will not highlight it as a prerequisite to be agile, while leanness will try to make the working processes flexible, but flexibility will not be emphasized as a prerequisite to be lean. Hence, the researches propose a combination of lean and agile methodologies, via the strategic use of a decoupling point, which works as a separator, with the lean processes in the upstream side of it and the agile ones in its downstream side. This creates the so called "leagility paradigm" (Naylor, *et al.,* 1999).

A leagile system has the characteristics of both the respective agile and lean production ones, acting together in order to exploit cost-effective market opportunities (Krishnamurthy, *et al*., 2007). The lean paradigm enables the upstream part from the decoupling point, where the demand is smooth and standard products flow through a number of value streams (Naylor, *et al.,* 1999), to be cost-effective (Mason-Jones, *et al.*, 2000a) and provide level scheduling (Banomyong, *et al.,* 2008). At the same time the agile paradigm enables the downstream part, where the demand is changeable and there is a larger variety of product per value stream (Naylor, *et al.,* 1999), to achieve high quality service (Mason-Jones, *et al.*, 2000a) and respond to the changeable requirements of the customers (Banomyong, *et al*., 2008). Consequently, leanness and agility, when used together, maximize the profits of a business (Banomyong, *et al.,* 2008; Mason-Jones, *et al.*, 2000a; Mason-Jones, *et al.,* 2000b).

Wang, *et al*. (2012) in their paper about leagile software development state the importance of Kanban for the improvement of the agile processes and try to explain the way this system works in terms of the leagile paradigm:

> Work should be "pulled" through the system as it is needed, as opposed to "pushing" it through. Only when a downstream process is ready and needs to do some more work does it pull work from an upstream process. The signalling between upstream and downstream processes is typically done via some sort of coloured card which physically travels between processes. The aim is to keep the process flowing at an even but continuous rate. This is achieved by controlling the number of Kanban cards which are in circulation within the process. (Wang, *et al.,* 2012, p. 1289)

The same scholars go further introducing Scrumban, a software production model based on both Scrum and Kanban approaches, used to maintain projects with numerous user stories and unforeseen programming errors.

> …. In such cases the time-boxed sprints of the Scrum model are of no appreciable use, but Scrum's daily meetings and other practices can be applied, depending on the team and the situation at hand. Using these methods, the team's workflow is directed in a way that allows for minimum earliest completion time for each user story or programming error, but on the other hand ensures each team member is constantly employed. (Ladas, 2009; cited in Wang, *et al.* 2012, p. 1288)

The flexibility and ability to easily adapt to market fluctuations, makes leagile paradigm attractive for organizations (Fogelström, *et al.*, 2010), yet there is little evidence that its applicability has been already tested in the real life organizations (Naim and Gosling, 2011).

Summing up what the literature reveals, lean production principles help software development industry to eliminate waste that might appear in different forms such as bugs, waiting time, unnecessary lines of code, etc. Kanban is the most common lean tool used by the organization nowadays to help them fixing defects and reducing work in progress. Nevertheless, lean philosophy does not enable the software development team to deal with the changeable user requirements, hence a combination of lean and agile approaches would benefit the organizations helping them to eliminate waste and manage change. This will guarantee them a better position in the market and as a result, a business value. The "leagility paradigm" ensures an integration of these approaches with the help of a decoupling point that works as a separator.

2.4. Managing people

The above-mentioned methodologies, that are associated with improvements to the overall business, will succeed only if the right people are supported and managed (organized, influenced and motivated) properly (Cawley, et al., 2010). Success in implementing and sustaining this paradigm involves three components of an organization: motivation, skills and environment (Eaton, 2003).

In software development projects, according to Chiang and Mokerjee (2004), because of the complexity of the customer quirements, the interaction of the client with the team during the development of the product is indispensable. To the same extent, it is crucial to

have a capable workforce, willing to share knowledge among team members and across various projects. Therefore, the benefits of technological improvement cannot be realized without the appropriate human capital (Chiang and Mokerjee, 2004).

2.4.1. Managing teams

Due to the human-centric nature of software development processes, special attention should be paid to managing, motivating and coaching the team members involved in software development projects. Blackburn, *et al*., (1996) state that people dominate tools and techniques, stressing the importance of customer specifications and their communication with programmers. They also realize that understanding the complexity of customer requirements, and dealing with other "soft" issues is more important than having to use tools effetively. Years later, Lindstrom and Jeffries (2004) confirm Blackburn's *et al*., (1996) point of view, affirming that the success of a project depends mainly on the people involved in it, more than on technology and approaches being used. Accordingly, it is essential to guarantee the appropriate working conditions and to influence and motivate the team members to achieve the highest productivity from them.

As mentioned above, when adopting agile, lean and leagile philosophies, a company has to go through continuous changes which are somehow challenging for the people involved in software development projects. It has been widely observed that a considerable part of the workforce tents to build in resistance to change due to different factors such as the style of management to which they were used before ("macro" or "micro") (Ewing, 1990; Huselid and Becker, 2011; Priem, *et al.*, 2011), their personality, individual competencies, etc. (Jones and Murray, 2008).

In some other cases, as in those of big-sized projects, the lack of direct contact with the customer might affect the engagement and motivation of the team members (Bosch and Bosch-Sijtsema, 2011). Therefore, management commitment and employee participation are important (Sim and Rogers, 2009). To the same extent it is important to be able to manage the emotions of the group members, especially those related to the feelings of frustration and optimism (Humphrey, 2002). Boddy (2001) and Maylor (2010) in their books about project management and managing teams, stress that behaviour is the key to project success. They go further explaining the importance of negotiating and influencing stakeholders up, down and across the organization, *in order* to gain their support for clarifying objectives, obtaining resources and monitoring and controlling.

Particularly, special responsibility have those people that lead the development team (Boddy, 2001; Maylor, 2010). It is important not only to help the team plan and coordinate to deliver the expected results, but also to motivate them to perform and go beyond the forecasted outcomes (Megerian and Sosik, 1997). Often employee intrinsic motivation comes as a special response to extrinsic factors, hence organizations should observe the preferences of their employees (Kreps, 1997). If they cannot use money to align incentives, then aligning preferences would be a practical option (American Economic Association, 2008). Milne (2007) points out that incentives, whether team based or individual, can positively affect the performance of an organisation as they enhance the motivation, performance and interest of the employees.

2.4.2. Emotional intelligence

"People skills" are vital and they should be continuosly improved in order to set and maintain the interpersonal aspect of the relationship with different stakeholder groups involved in a project (Riggio and Lee, 2007). As Jo and Shim (2005) affirm, support and communication can increase job satisfaction and employee performance.

Another concept closely related to "people skills", is the one of "emotional intelligence". Emotional intelligence is defined as the ability to perceive, appraise, express, use and manage emotions in order to achieve an efficient decision-making process (George 2000), to develop cognitive collective goals and collectively set their priorities and to motivate team members by generating trust, confidence and enthusiasm (Groves, 2006). A key part of emotional intelligence is empathy, which consists on the ability to perceive the thoughts, emotions, or experience of others and it is positively related to job performance (Gentry and Weber, 2007). Therefore, people leading software development teams should be aware of their team members' feelings and emotions, should motivate and reward them for the successful attempts. They should also try to delegate authority when their followers know what to do and are motivated, knowledgable and committed to do it; they should instead coach (direct and support) them when their competence and commitment in certain circumstances are fairly low (Geller, 2000).

2.4.3. Managing top-down

Another important factor to be considered when aiming to increase productivity, is the close and direct cooperation between top management and team members in order to

achieve lower prices, lower costs, higher, profits, higher wages and a higher level of competitiveness in the market (Emiliani, 1998). To observe to what extent are literature remarks apply to real life businesses, all the interviewed individuals where asked about the soft aspects of management and the practices that they follow in order to motivate, influence and coach their respective teams.

Summing up, the literature suggest that human capital is more important than the technological one, since people run the working processes to deliver the expected outcomes. In order to get the best out of them they should be influenced, supported and motivated. Therefore, managers need to take a “servant-leader” role, to deligate authority and to be emotionally intelligent when managing the respective teams in order to increase their productivity.

3. METHODOLOGY

The literature review chapter is mostly written based on journal articles and conference proceedings found in the online databases of the libraries. As Powell (2008) observes, appropriate and recent literature sources (preferably published within the last five years, unless the work has long-standing influence) offer innovative contemporary insights into the topic.

After having completed the literature review that serves as a theoretical base for the current study, the focus of this chapter will be on studying the methodology used to carry out this research. Methodology is defined by Gelo, *et al.* (2008) as "….as a set of rules, principles and formal conditions which ground and guide scientific inquiry in order to organize and increase our knowledge about phenomena." (Gelo, *et al.*, 2008, p. 270)

The two main methodologies in the academic world are the quantitative methodology and the qualitative one (Newell, 1992; Duffy, 2005; Mahoney and Goertz, 2006; Sobh and Perry, 2006; Gelo, *et al.*, 2008; Polit and Beck, 2010). A piece of academic work, no matter the topic that it is written about, involves several phases such as problem formulation, data collection, data analysis and reporting (Soydan, 2002). Quantitative and qualitative approaches apply different research methods with reference to most of the above-mentioned phases (data sampling, collection, analysis, and interpretation) (Gelo, *et al.*, 2008). The next section provides an explanation of both qualitative and quantitative approaches, in order to provide the reader with information that helps to understand the decision of the author to use only one of these approaches for the purpose of the study.

3.1. Qualitative vs. Quantitative

In order to answer the research questions, data should be collected and analyzed; the way of going through these processes defines whether the approach is qualitative or quantitative (Gelo, *et al.*, 2008). Quantitative approaches rely on numerical values (Mahoney and Goertz, 2006; Gelo, *et al.*, 2008) and large samples (Sobh and Perry, 2006) to carry out statistical analysis (Mahoney and Goertz, 2006; Gelo, *et al.*, 2008) and to test theories (Sobh and Perry, 2006). By contrast, the majority of the qualitative research is based on non-numerical data collection (Gelo, *et al.*, 2008). This kind of research aims to understand the human experience by studying intensively particular cases (Polit and Beck, 2010). In fact, they start with cases and then move backwards to the initial causes adopting a "cause-

of-effects approach" to the explanation (Mahoney and Goertz, 2006), trying to use the meanings found in small samples to build theories (Sobh and Perry, 2006). Therefore, generalizing from particular to broader constructs is mainly related to qualitative approaches (Polit and Beck, 2010). Cachia and Millward (2011) cite Kvale's (1983) work where qualitative research is defined as an approach that consists basically in conducting interviews, for the purpose of gathering descriptions of the life-world interviewee, by focusing on specific situations, actions and consequences, in order to find out more about the phenomena that is being observed.

Many researchers suggest to chose only one type of methodology and apply it to a study, as using both types might be inappropriate, or produce contradictory and/or unrelated findings about the same phenomena (Sobh and Perry, 2006). For the purpose of this study, qualitative methodology is used to collect and analyze the data in order to find the answers to the main research question. This is because the study is based on the literature review, to find out the phenomenon that would be interesting to conduct research upon. Afterwards, the experience of the participants, not numerical data and statistics, was used to find out more about the topic under observation.

3.2. Research aim

As mentioned above, the process of conducting research on a certain topic goes through different phases such as problem formulation, data collection, data analysis and reporting. (Soydan, 2002).

The problem formulation, as Soydan (2002) claims, is compounded by:

- A practical issue - determines the possibility of conducting research on a certain question, the accessibility of the data, the ethical issues to be considered etc.)
- In a scientific process - determines whether a theoretical base and an appropriate methodology for the research topic exist. (Soydan, 2002)

In the current study, the main issue is to define optimal ways to improve the productivity and increase business value of the software development projects. Consequently, the accessibility of the data is mainly influenced by the possibility to find literature (a theoretical base) related to this question, and people who have a background and/or practical experience to give their contribution towards the current research. Ethical issues are considered but those will be discussed later on, in this chapter.

3.3. Research Design

A *research method* includes "those procedures and techniques involved in data collection, analysis and interpretation" (Gelo, *et al.*, 2008, p. 274), whereas a *research design*, consists in "a plan of actions or structure which links the philosophical foundations and the methodological assumptions of a research approach to its research methods, in order to provide credible, accountable and legitimate answers to the research questions" (Gelo, *et al.*, 2008, p. 272). This plan covers who is studied, design considerations, the time frame for data collection, and the threats to reliability and validity (Hernon and Schwartz, 2009).

Since the qualitative approach is used as the only research method suitable for the purpose of this study, data collection, analysis and interpretation are totally based on this methodology. The whole study aims to answer the main question: "How can the productivity of software development projects be improved in order to increase the business value in software development industry?" The theoretical groundwork is connected to the methodological assumptions by creating a concrete research design, which consists on the following steps:

- Defining the rationale for the study
- Reviewing carefully the literature related to the topic, to find out more about the previous research conducted in the field, and to create a supportive framework for the current research.
- Creating a research project and identifying the target group, from which a sample of individuals will be chosen to participate in the study.
- Identifying gaps in knowledge between theory and real-life experiences
- Giving recommendations on how to improve productivity, eliminate waste and increase business value of the software development projects.

The data collected from those that have been subject of this study, whether by being interviewed or answering questionnaires, has been gathered in a timeframe of a two months. The short time of observation might influence to a certain extent the reliability of the results.

3.4. Data collection

In order to collect data for a study it is important to create a representative sample. This can be done by selecting the individuals randomly, so each member of the population has

an equal chance to be included in the study (Polit and Beck, 2010), avoiding every opportunity for selection bias (Newell, 1992). In this way, it will be easier and more reasonable to generalize the results of the research, even though sampling and generalization might be in conflict sometimes (Keppel and Zedeck, 1990). For the purpose of this study, the individuals for the interviews were chosen based on convenience sampling[8], which means that only Scrum masters were interviewed to find out to what extent agile and lean philosophies have improved the productivity of the software development projects. At the same time, the selection of the individuals to fill in the questionnaire has been very random.

The definition of the sample leads to the data collection process. The data can be either primary (extracting information by the members of the sample by using interviews, questionnaires and closed-ended observational protocols), or secondary (referring to other official documents and research archives) (Gelo, *et al.*, 2008). The primary data sources are direct, whereas the secondary ones aim to make a noticeable development from the original (Powell, 2008). As for the interviews, theoretically they can be divided in three main groups: structured semi-structured and unstructured (Carruthers, 1990).

The participants of the study were selected in several ways. The first has been by going through various blogs in online professional networks, reading certain discussion on topics related to software and product development productivity, agile and lean philosophies, lean software development, etc. and finding out the most active, zealous and knowledgeable participants in the discussions. Afterwards, a personal message was send to various individuals explaining the purpose of this study and the benefits that their contribution would bring to it. A second way has been by contacting companies, introducing the topic and explaining the situation. In some cases, some of the current participants helped in establishing contact with other professionals of the field who would be willing to give their contribution on this research. Two of the interviewees are working in the same project; two others are representatives of the same company, working on different projects though; and all the rest of the interviewees represent different companies, operating in several countries of the world.

Since this study is based on the qualitative methodology, triangulation is used for the collection of the information and it is based on primary data. Triangulation, a synonym for

[8] "…a sampling in which elements are drawn from a subpopulation according to its accessibility and research interests" (Gelo, *et al.*, 2008, p. 275)

mixed methods (Bazeley, 2004, cited in Sobh and Perry, 2006), means that data is collected from multiple data sources (Crandall and McGaghie, 2001), using a variety of methods (Guba, 1980; cited in Carruthers, 1990). Concretely, the collection of data is attained by conducting semi-structured and open-ended[9] interviews with eleven Scrum masters, and an online questionnaire, which was answered by several Scrum masters and practitioners worldwide.

Interviews are used as a research method since they are an appropriate type to collect information for the qualitative approaches, and help to share knowledge and experience between pairs by having the form of a professional conversation (Kvale and Brinkmann, 2009; cited in Cachia and Millward, 2011). Semi-structured because they contain elements of both structured and unstructured interviews and have the type of a managed conversation, compounded by a fixed set of principal questions that can be used as an "interview skeleton" and additional follow-up questions to facilitate the exploration of certain issues that the interviewee comes up with (Cachia and Millward, 2011). Semi-structured interviews benefit the study by helping to collect the necessary data, matching certain questions to specific interviewees, while reducing redundancy and maximizing comprehensiveness at the same time (Gugiu and Rodríguez-Campos, 2007).

In the current research, six out of eleven interviews were audio-recorded, and transcribed, while each of the other five participants sent in their respective information by email. The first group of interviews was realized by synchronous communication, more concretely by meeting the interviewees in person in four cases, and by the help of technology, realizing conference calls, in two other cases. The participants met were representatives of big successful companies with branches in Gothenburg, with a huge experience in both software and product development industry. The participants that were reached through internet were representatives of the same industry, one of them in Europe and the other one in Chicago, USA.

The questions were sent in advance to most of the participants to give them the opportunity to reflect about them and prepare the answers. The interviews were transcribed as soon as possible in order to preserve accuracy and gain a different view of the same interview while reflecting upon and analyzing the data being transcribed. This is a practice that

[9] Open ended interviews are compounded by questions that are formulated in a format that requires a verbal, or a narrative reply (Duffy, 2005)

McBride (2008) suggests in his paper when applies almost the same methodology to analyze software development projects.

Each of the interviews had a duration of between 25 and 90 minutes. The interviews were compounded by eight main questions drawn by the literature review and two other questions developed by the author of this study.

The second form of data collection used in this study, the online questionnaires (which nowadays are becoming increasingly popular (Cachia and Millward, 2011)), was used to overcome the perceived weaknesses of a single-shot approach (Wood, *et al.*, 1999). The questionnaire consists in a set of questions which are combined under certain subtopics. For the purpose of the study, some of the questions are structured (multiple-choice questions or questions that require a yes/no answer), or unstructured and open-ended ones used with the intention of giving the questionnaire filler the freedom to express him/herself and share his/her experience and insights about the topic. The questionnaire contains questions somehow similar to those of the interviews, and it was done to make the study more complete, by adding the experience of other professionals who did not have enough time to undertake an interview.

The survey was published online in online professional group pages and was answered by representatives of software development industry throughout the world (Canada, Brazil, India, USA, China, Ireland, Poland, Ethiopia, Sweden).

3.5. Data Analysis and interpretation

Data analysis consists on analyzing the information gathered to address the research question or hypotheses; more concretely qualitative data analysis is based on the content and systematic analysis of the text data gathered during the data collection process (Gelo, *et al.*, 2008). The collected data should also be interpreted in order to find a meaning to the findings.

In the current study, the results are processed after the transcript of the interviews and the collection of the online filled questionnaire. Especially in interviews, due to the nature of the questions that aims to allow participants to share as much information as they want to, the interviewees often provide a large amount of information, part of which might be out of the focus of the main purpose of the research. Therefore, after the collection process, data

is elaborated and reduced, in order to fit better the needs of the study. Then the findings are analyzed, interpreted and judged in terms of reliability and validity (refer to the last section of this chapter).

3.6. Ethical considerations

Every study that involves human subjects should meet the standards of ethical behavior when conducting research and publishing the results (Regehr, 2001). This study does not include any sensitive information as the participants were asked general questions that were mostly based on their personal views and experience. However, the interviewees were introduced in advance with the main aim of the study, the subject and very often the questions of the interview. This was done initially by an introductory email sent to each of them, and later on by a short pre-interview session consisting in a few minutes of the study's topic, aim and objectives representation. By the end of the interview, each participant was asked to sign a consent form in which they confirm that:

- they are aware of the research aims,
- they give their permission to audio-record the interview
- they had a chance to talk about the ethical concerns of the topic (i.e. the anonymity of the organizations they refer to in concrete examples, any sensitive details about certain projects, etc.)

In addition, the participants were told in advance that no concrete details such as the name of the organizations, the funds invested in certain projects, etc. are needed for the purpose of this study. In case they would come up with such information, it would not be used and published in this dissertation.

3.7. Limitations

The results of the study should be reliable, valid and trustworthy taking in consideration the strengths and limitations of the research and judging about the generalizability of its findings (Regehr, 2001). External validity is the degree to which the findings may be generalized to other contexts (Leedy and Ormrod, 2001; cited in McBride, 2008). Due to the fact that this study includes only 23 people (11 interviews and 12 people who have answered the survey), the results cannot be considered generalizable but of course, they might be seen as helpful and constructive for the research that might be conducted in this area in the future.

4. RESULTS – FINDINGS AND DISCUSSION

This chapter, which will be divided in several sections, includes an analysis of the data gathered through interviews and questionnaires. They were conducted in function of the main goal: to identify potential ways to measure and improve the productivity of the employees involved in software development projects and to observe whether agile and lean methodologies have had a role in the productivity of the working processes or not. The ways to measure and improve productivity, and the importance of agile, lean and leagile paradigms are based on the views and the experience of the participants involved in this study.

The rest of the participants are 12 people that have filled out the online questionnaires, 10 of which with more than 10 years of experience in software development projects. Eleven of these people have been involved in agile projects, and only six are experienced in working with lean software development projects, too.

4.1. Productivity measurement and improvement

The majority of the participants state that measuring velocity from one sprint to another is an effective way to assess the productivity of the development team. The velocity can be measured in relative story points per day, function points per hour, etc. According to one of the participants, this is calculated across various projects periodically (once in a month, once in three months, etc.) and reflected in graphs to see the trend of the productivity. Measuring productivity based on the number of function points aligns with Symons' (1988), Maxwell's and Forselius' (2000), and Ferrão's (2010) studies, which retain function point counts as an optimal way to measure productivity. However, these scholars place importance on lines of code per programmer (Collofello, *et al.*, 1983; Scacchi, 1995; Maxwell and Forselius, 2000; Stroustrup, 2012) as an optimal measure of productivity. It can be easily seen that counting the story points instead of counting the lines of code per programmer has resulted to be more effective nowadays. The participants do not mention pages of documentation per programmer (Collofello, *et al.*, 1983), implemented changes (Petersen, 2010) and other measures brought to attention by the previous scholars, as ideal productivity measures.

They instead claim that measuring a team's performance to their performance in a moment earlier in time, is another way to observe to what extent the productivity of the work

processes is improved. Some interviewees claim that it is important not to compare the performance of two teams as long as they are not working under the same conditions.

The number of features delivered over a specified period and the length of time it takes a feature to go through changes are other measures of team productivity. One of the participants argues that the length of the transition time that it takes a feature to go from "in progress" to "done" is a better indicator than counting bugs, hours, and comparing estimates to actuals (S. Fenton, email, July 7, 2013). He seems to oppose the thoughts of other participants who give importance to the comparison between the planned and the actual activities (in terms of time, cost, person-hours/weeks, etc.), the variance and the earned value analysis (NK Shrivastava, personal interview, June 23, 2013). In a certain way, his view also contradicts the statement of another interviewee who considers as an important measure of productivity the time needed to fix a bug as it shows the maturity of the workflow (V. Thavonekham, personal interview, June 27, 2013)

Other organizations measure the team performance by monitoring to what extent the milestones initially set are fulfilled.

> We have a process to set several milestone for a work package (or product development life cycle): preparation, formal start, commitment made, iterations (1-N) done and closed. For each stage, there are some checklist to make sure all the criterion are fulfilled (L. Zhang, email, June 13, 2013).

According to another interviewee, observing and measuring quality is a way to measure the performance of the team.

> ….a way is to measure the quality. Quality is based on three parameters: the level of rework (the repair build ratio – how much time you spend in repairing a task that took a certain amount of time to be accomplished; normally it should be less than 5-15%). It is important to point out that the number of defects is not an objective measure, as when you compare two teams that might have gone through the same number of defects, it is difficult to compare the level of the respective problem sizes. The second parameter of quality is change. If the initial requirements are clear, there should not be any changes to slow down the progress of the work. While the repair ratio reveals the quality of work done by the development team, the change ratio reveals the quality of the requirements produced. These measures are reflected in a score card, and the score card is placed on a dashboard, which is available to everybody and helps to measure each month the performance of the team. (NK Shrivastava, personal interview, June 23, 2013)

In addition to the above-mentioned measures, there are soft values as well. As such, it might be worth to mention the intuition of the team leaders; the way they feel about the work accomplished by the team members (S. Singman, personal interview, June 17, 2013). On the other side, according to one of the participants, the mood of the team reveals the problems earlier than statistic and metrics do. If the moral of the team drops and nothing is done to improve it, it will be reflected in longer cycle times and slower working paces. Therefore, it is necessary to collaborate closely with the team and observe the changes (S. Fenton, email, July 7, 2013).

Almost all the participants of this study retain customer satisfaction as a crucial factor to be considered when aiming to improve productivity because happy customers tent to result in business value increase. This can be achieved by establishing a steady contact with the customer during the sprint in order to develop and modify the code aligning it to the customer needs and letting the customer interfere and provide feedback before the final release.

The reflection phase about the lessons learned at the end of each release is judged to be crucial to the productivity improvement according to one of the participants of this study. It can be easily seen that the measures of productivity that the participants of this study provide are somehow different from what the literature suggests. Several researches have defined productivity as the functionality, complexity and quality of the software products (Anselmo and Ledgard, 2003), or as the time and cost required to deliver these products to the market (Duncan, 1988). In the concrete study, as observed above, quality is a key factor to be considered, but besides, others are the productivity measures that the participants come up with.

Despite their personal working experiences, the participants of this study retain it difficult to measure the productivity when aiming to assess efficiency, in order to increase the business value. This supports Chiang's and Mokerjee's (2004) statement that the intangible nature of software makes it almost impossible to measure.

4.2. Software development life cycle

When it is to the length of the development cycle participants have different opinions regarding its impact on productivity. To a certain extent they agree on the fact that shorter development cycles are the best as they translate in less time to get to the customer; in

other words, in faster delivery, in earlier feedback and as a result in improved business value. They help to check continuously if the milestones of a project are being met and to change direction if the results produced are not the expected ones. One of the participants adds to the above advantages by pointing out that a shorter development cycle also translates in earlier return on investment.

> If you build the most valuable functionality first, and release it as soon as possible, that functionality can be earning money as soon as possible. Often the value of this first release is enough to offset the cost of the rest of the project so in such a case you have recouped your investment in a month rather than in a year. In addition, by shortening the cycle time, without reducing the size of the increment, you are doing the same amount of effective work in a shorter time, thus by definition, as it were, increasing productivity (P. Oldfield, email, June 28 , 2013).

Despite the benefits of shorter development cycles, they are often associated with challenges. The team who decides to produce code everyday can be quite productive, but it has to be mature and to have an excellent planning capacity to organize the tasks (V. Thavonekham, personal interview, June 27, 2013)

On the other side, long development cycles require more planning efforts and are associated with the risk of running too far in the wrong direction if no controls are exerted in a long time.

What is retained as optimal is a development cycle with a length between two and four weeks, in more specifically, a three-week cycle results to be the most efficient one in the majority of the interviews. However, the duration of a cycle depends on the complexity of the feature and on the existence of the other teams (in case of large projects with dispersed teams) throughout the world with the purpose of coordinating and aligning the activities, in order to ensure an effective collaboration. Related to this issue, V. Thavonekham provides an interesting example during a personal interview on June 27, 2013.
Regardless of the above views, there are other participants that claim that faster development cycles do not necessarily result in better quality and increased productivity.

> Projects with faster development cycles are not necessarily more productive. I mainly think about projects in which the quality of the stories is not high enough. I also think about projects where the framework is badly put together. Some agile projects seem to postpone the thinking step until after the development. It sometimes might look like such projects quickly deliver working software. However, after some sprints, as the software gets more complex, it sometimes results in the realization that wrong decisions were made, and un-doing this sometimes slows down a project. A possible side-effect is that teams keep on fine-

> tuning and re-factoring code instead of developing new functionality (T. Vermeerschen, email, July 3, 2013).

None of the scholars, whose work is included in this study, focus the attention on the negative effects of executing very short development cycles. Contrariwise, Blackburn, *et al*., (1996) argue that the customers will not wait for the product delays; the delays can be destructive as the software market offers various substitutes.

Other researches such as Livermore (2008) and Cordeiro, *et al.* (2008) retain that the optimal length of a sprint is from one to four weeks, a statement that can be proved to be true in practice when judging the answers of the individuals involved in this study.

4.3. Team size

When discussing about the ideal team size, all the participants of this study mention that theoretically it would be good to have teams compounded by seven +/- two people. The ideal team size slightly varies from one interviewee to another. Some agree on 4-8 people per team, other to no more than seven, others on 6-9, while two other ones see eight as an effective size. Despite the small variances, all the interviewees assert the advantages of middle-sized teams, stating that a team should not be too small, but not too large either. This aligns with is the study of Schwaber and Sutherland (2011) who see the middle-sized teams as the most efficient ones. According to the participants of this study, in the case of very small teams, it is unnecessary to embed Scrum and follow its artifacts because it slows down the working processes. On the other hand, teams compounded by only a few individuals do not have an effective discussion, as it is not possible to provide many different angles to brainstorm and judge certain situations during their everyday work. Schwaber and Sutherland (2011) state the same in their study when they claim that very small teams decrease development speed and productivity.

Nevertheless, one of the participants points out the advantages of smaller teams, bringing to attention that they can be more easily self-regulating and self-organizing, improving noticeably the development speed. This point of view aligns with the concept of “tiger teams” brought to attention by Blackburn, *et al.* (1996). Another participant comes up with a concrete example in order to show the impact of adding members to a team:

> ….We recently added two new team members. Initially, this seriously reduced the flow of work. The long-term effect was an overall increase. Whenever you change the line-up you will take a hit in this respect whether you are adding or removing

> team members. It is hard to give consistent results with less than 4 people and different tactics are required when the team gets bigger than 30 people (S. Fenton, email, July 7, 2013)

Summarizing the interviewees' thoughts, in case of large teams, they might find it difficult to be autonomous and self-organizing due to the large number of the team members. The participants point out that the team size and the number of communication channels are directly connected with each other. One of them goes further providing the formula that reveals the dependency between the team members (N) and the number of communication channels created (M): M= N*(N-1)/2 (NK Shrivastava, personal interview, June 23, 2013). A large number of team members causes too much overhead within the team and leads to the creation of small teams within the big team decreasing team cohesion and leading to a lower productivity. Another difficulty that face large-sized teams is finding a room to fit in. The participants state that it is beneficial for the team to be collocated, as they find it easier to communicate and collaborate. Again, their experience is supported by the study of Schwaber and Sutherland (2011), which claims that very large teams (those with more than nine members) require special efforts too coordinate and manage their work.

Another participant prefers to skip discussing about the ideal size, pretending that the adjustment of the team size depends on the volume of work that needs to be done. He states that the team should include that amount of competent people that possess enough of the skills between them to form a viable group that will be able to cope with the complex requirements of the job.

Special importance to the team skills has been placed by Chiang and Mokerjee (2004), who claim that a team compounded by skilled individuals can improve the development speed and productivity; it also can mitigate the schedule risk associated with the development process.

4.4. The role of agile approaches in improving the working processes

When being asked whether agile methodologies have improved the working processes, a considerable part of the participants, agree on the superiority of these approaches over the traditional ones. Operating agile translates in faster delivery, continuous interaction with the customer and higher quality of work, changing the company's mindset and helping it to resolve impediments and leverage success. All these advantages brought to attention by the participants fully agree with previous research of Ambler (2009), Mishra and Mishra

(2001) and Rico (2011) who stress the advantages that these methodologies have brought to the software development industry. The researchers go even further claiming that all these positive sides of agile methodologies improve the organizational morale of those enterprizes that implement them.

One of the questions of the online questionnaire was ”To what extent have agile methodologies improved the software development processes?” where the respondents were asked to choose among:

- No improvement at all
- Slightly improved
- Huge improvement

None of the respondents thought that agile methodologies have not improved the working processes; instead 5 out of 11 chose the second option, whereas six others chose the third. Perhaps the reason why five respondents see agile as beneficiary up to a certain level can be found in an interview where one of the interviewees claims that agile does not necessarily increase the development speed, but it improves the quality and helps to deliver value to the customer (J. Hermann, email, July 2,2013). According to another interviewee, using agile allows fine-tuning the processes, methods and the way of working (T. Vermeerschen, email, July 3, 2013). Those teams that operate agile will be able to cope with changes and do the retrospective and pursue continuous improvement (L. Zhang, email, June 13, 2013). Embedding agile methodologies makes the organizations turn into “learning organizations” and integrates the development team into the process as a whole. One of the participants provides a concrete example from his own working experience:

> Agile methodologies have improved a lot the working processes. I have a very good example of a project on which we were working some times ago. We used to work with waterfall, and we were basically too late to the market, we did not hand in to the customer what they wanted, and when we gave them what they wanted, they had already changed their minds, because internet came up in this era and everything should be web-based instead of just a piece of software. So first, we were too late to the market, and second, we didn’t get enough customer feedback during the process. Furthermore, we had huge integration problems. Then we started to work with Scrum in a very small project that was specific to one country; we had the quick iterations, sprints, we got the customer feedback all the time. We saw the advantages and we noticed that we could implement that in bigger projects. So we started to implement it in a very large project in India, Sweden, Norway, Germany and U.S., where the implementation teams were located. There were in total twenty-five Scrum teams at the edge. It was very, very

> effective as we increased not only productivity, but we delivered also the right things and the quality went up dramatically. So by the end we delivered a much, much better software. To sum it up, agile development methods showed to be able to improve the delivery time, the interaction with the customer and the quality of the work (M. Jonsson, personal interview, June 25, 2013).

All the articles that have been written about agile methodologies and that have been used to build the theoretical groundwork of this study, stress the superiority of the agile approaches over the traditional ones. However, not all the participants of this research agree on this point. According to two interviewees, it is useless to compare traditional and agile methodologies as depending on the situation they are both useful and beneficial to the development team. They are two different approaches, both strong in their own environments. Using agile would make a difference only in those teams that are continuously dealing with change. If the customer requirements are clear from the very beginning, traditional methods such as waterfall, would be effective to the same extent as the agile ones (M. Gustafsson, S. Singman and NK Shrivastava, personal interviews, June-July, 2013). Indeed, as it has already been seen in the literature review part of this study, traditional methodologies are suitable on those projects characterized by static business requirements (Northover, *et al.*, 2007; Livermore, 2008; Ruparelia, 2010). Even though the literature states that traditional methodologies are overloaded with documentation and procedures, it looks like this is not a concern for those participants of this study that do not make a difference between traditional and agile approaches.

Previous scholars, such as Ambler (2009) and Rico (2011), when pointing out the advantages of implementing agile, focus the attention on the flexibility of the structure and the visibility of the stakeholders that lead to effective risk management and decision-making processes, advantages that surprisingly are not mentioned by the participants of this study.

4.5. Moving a team to agile

Agile methodologies, as seen in the literature review section, benefit the results of the development team in a lot of ways, therefore they are seen as crucial to the success of software and product development projects. It is obvious that for small projects, the agile approach has to be fully implemented, but what happens for the large-sized projects that are compounded by various teams? Do all of them need to be on agile methods in order to

be productive and achieve highest results? According to the opinion of the people that contributed to this study, it is vital to implement the agile approach in all the parts of an organization, no matter the function of specific individuals. Agile methodologies, according to the majority of the participants, help to deliver more code, in a shorter time and with a higher quality. An additional benefit associated with them is that employees get to learn more by being involved in cross-functional teams. All these advantages brought to attention by the interviewees confirm the theory of Amber (2009), Fogelström, *et al.* (2010), Mishra and Mishra (2011), and Rico (2011) that when listing the advantages of agile methodologies, mention the faster time-to-market, the lower costs and the higher quality of those software project deliverables that have been run on agile methodologies. As for the cross-team communication, it ensures an effective decision-making process, as it allows input from various teams and synchronization among them due to the dependencies that they have with each other (Mishra and Mishra, 2011).

The interviewees claim that not necessarily all the members of a team need to be moved to agile; however each of them should be knowledgeable of agile principles and should have some kind of role within the agile team. Otherwise, as S. Singman (personal interview, June 17, 2013) states, it might result in coordination and transaction overhead with can turn out to have high costs for the company.

The interviewees state that moving the right people to agile is a long process, associated with many challenges. Perhaps one of the biggest ones is convincing people to get out of their offices, share the same space and collaborate by sharing their knowledge with their colleagues. Agile is a totally different mindset, that changes the organizational culture. Depending on the way people are used to work, on the management style that they are used to, and somehow on their personality traits they might find it frustrating and challenging to deal with this change.

> If you have get used to and also play quite well in the old way, it takes time to make people accept new things. Comparing to the old way, people will be more involved in the agile, faster reaction, daily status check in the daily meeting, etc. will bring more stress and pressure to some developers. Short pain is inevitable but when the team becomes a experienced agile team, they will definitely love it and use it (L. Zhang, email, June 13, 2013).

> The only drawback is that when individuals of the big organizations are brought to work together in Scrum teams, some of them find themselves too challenged, mostly emotionally, as for example when they lack the necessary skills to keep up with the pace of the others (A. Sandberg , personal interview, July 3, 2013).

Indeed, this confirms the studies of Chau, *et al.* (2003) and Nerur, *et al.* (2005) who recognize the tacit nature of knowledge that has to be exchanged among the members of agile teams and the difficulties associated with this process. Participants think that this might turn out to be a problem especially in large projects due to the complexity of the tasks, the large number of the stakeholders and the large amount of the information exchange, confirming Mishra's and Mishra's (2011) corncerns about this issue. When asked about the potential challenges associated with this process, one of the participants answers:

> The challenges? Many. Very many. People see it as taking away their power; people see it as undermining the knowledge they have built up over the years; people don't like to have to think; people think it doesn't apply to them; people find it really hard to change even when they want to, because they have forgotten why they do things the way they do, so they are afraid of what will break if they change... That list covers just some of the major problems, and each of these will manifest in many ways. Then you get the keen ones - the ones who want to throw away all they know and get stuck in to a "new and better way of working" without understanding the problems this will bring, not just for them but for the project as a whole. When you are already agile, you can deal with each of these problems as they arise, but while you are still learning, while changes in approach and tweaks to the process are still painfully slow and difficult, each of these problems has the potential to become big and serious (P. Oldfield, email, June 28 , 2013).

This point of view goes in parallel with Middleton's and Joyce's (2012) thinking, who claim that the organization should go through huge change management initiatives in older to deal with the new methodology and change the organizational culture towards implementing the new phylosophy of work. Perhaps a good way to overcome this challenge is by providing examples of cases when implementing agile resulted in successful outcomes, in order to engage employees and involve them emotionally. From the experience of the participants, it seems like nowadays organizations use the help of the agile coaches to train all those individuals that have to move to agile.

> ..... We did a mistake years ago as we didn't undertake any training, so we basically trained ourselves and it took us some time to get used to work with Scrum. If we would have agile coaches, as we nowadays do, we would experience a faster and a more effective process of moving from waterfall to Scrum (M. Jonsson, personal interview, June 25, 2013).

Another way of moving people to agile is by implementing it in a small team in order to observe the results and judge whether it is worth to expand its implementation or not.

> ..... It is recommended that you start working agile with a small isolated collocated team, and achieve some small, measurable successes. That evidence then promotes

> further expansion of agile to other teams, eventually to the entire development team (J. Hermann, email, July 2, 2013).

When moving a team to agile, except from the psychological resistance to change, additional challenges that the participants bring to attention are:

- Top management involvement in supporting the organizational change;
- Managerial expectations of command-and-control reporting;
- Lip-service managerial support without follow-through;
- Incompatibilities between agile team and overall traditional organization especially at touch-points
- Influences from outside the team (i.e. resources being removed from the project while unannounced, the sprint backlog being modified while the sprint is ongoing, etc.),
- Doubts on what should be tested inside and outside the sprint,
- "That's not my topic" attitude, etc. (T. Vermeerschen and J. Hermann, personal interviews, July, 2013).

The resource accuracy issue, the complexity of the domain, and the support of top management were mentioned previously by various researchers, such as Gunasekaran (1999), Chau, *et al*. (2003), Nerur, *et al*. (2005), Mishra and Mishra (2011) and Wang, *et al.* (2012). Different from literature that brings up the complexity of the information infrastructure for the dispersed teams as a challenge (Gunasekaran 1999; Bosch and Bosch-Sijtsema, 2010), none of participants mentions this factor, even though they recognize the difficulties associated with complex infrastructures when discussing about the ideal length of the development cycle.

4.6. Agile and Lean integration

The majority of the Scrum masters that participated in this study are knowledgeable about the benefits of embedding lean principles in agile methodologies, but in practice the majority of them do not have enough working experience in using the leagility paradigm in concrete projects. This is the reason why some of the participants of this study are not able to judge whether Scrum, Kanban, or scrunban would suit better to their specific working processes. Nevertheless, one of the participants, after making the distinction between lean and agile approaches, sticks to lean claiming that it is the best approach since it can encompass the whole organization:

> While Lean and Agile have a great synergy, there are two specific areas where they differ, and I have to say that in the ideal world, "Lean" has the better approach in both cases... but few of us live in the ideal world. Lean asks us to consider the whole process. Agile separates out the systems development part, and in some cases deliberately isolates the development team from the people working on the rest of the process. The other major point of difference is that Lean expects the Manager to be an expert in the process; to manage the work by managing the process. Agile has an unwritten assumption (nevertheless often true) that the manager will be ignorant of the nature of the development process and managing the process is best left to the development team.
> In short, "Agile" in its more common forms is designed to allow agile working where the organization as a whole is NOT lean or agile; Lean is designed to encompass the whole organization - and where it does, a fair proportion of the common 'agile management' aspects of agility are out of place (P. Oldfield, email, June 28 , 2013).

However, such a difference is not pointed out by any of the scholars whose work is used as a framework to support this thesis, so it is interesting to observe the difference between theory and experience at this point.

Some of the Scrum masters interviewed have some experience in using both lean and agile approaches. Nevertheless, they do it in different teams within the same project, since they find it difficult to embed lean and agile within the same team. They assert that contemporary agile projects combine Scrum and Kanban for the benefit of their working processes. Other Scrum masters, with working experience in both lean and agile approaches, state that while Scrum serves to develop new features, Kanban helps correct the defects after software is released and feedback is received from the customers.

> …On my opinion, Kanban would be suitable for the “after market”; once you have delivered the software, you it helps to look after defects, small enhancements and so on. If you are familiar with Scrum and you have some knowledge about how Kanban works, it would be easy to integrate Kanban processes, especially to handle defects. If you have a team that is delivering sprints in a project and, after they release them, they get some small defects on the product, you can fix them using Kanban. The ideal would be to have several teams within a project, let’s say a total of six teams for example, five of which will focus on developing the software; then you take one team member from each team and put them in a sixth team. In this way it will be a team of five persons that will work only on defects. This means that the other teams can continue to work on new development and new development only, and at the same time you have a team working in parallel with them to fix bugs (M. Jonsson, personal interview, June 25, 2013).

> ....There are different versions of a product. The Scrum and Kanban teams work at the same time, but while the Scrum teams work on the new versions, the Kanban team works on the older ones correcting the faults and errors. (M. Gustafsson, personal interview, June 14, 2013)

> ....An organization can have both agile and lean mindsets. They do not contradict each other. Lean is about getting things done as soon as possible, it is about continuous improvements. (S. Singman, personal interview, June 17, 2013)

Referring to the literature review part of this study, one can easily realize that the above statements confirm Poppendieck's and Cusumano's (2012) findings about the way these approaches work

Of course embedding both lean and agile methodologies within the same organization is often associated with challenges that are mostly related to the nature of the lean philosophy that aims to eliminate waste, and with people's reluctance to eliminate part of their work, no matter if it does not add value to the business.

> ...To implement lean means to remove the low-value activities: to look at each activity done for the project and to do a value stream mapping, assign a value to each step of the process, identify those activities/processes that have the lowest value and remove them. Lean principles can be applied in all the methodologies (traditional and agile ones). However, identifying and removing the low value activities is a real challenge. Mostly, people do not want to remove anything. Besides, they are not very familiar with measuring value so this is also a challenge in a certain way. (NK Shrivastava, personal interview, June 23, 2013)

As for the "leagility paradigm", it seems to be a relatively new concept in the organizations represented by the participants of this study. For several reasons (i.e. the lack of infrastructure; the conviction that developers should focus on what they are doing without getting distracting in additional things, the time-constraint deadlines, etc.) the organizations have not implemented it yet, so the whole picture of the way this paradigm works is explained only in theory.

4.7. The connections among people-technological processes-principles

The individuals involved in this study see people, technology and approaches as crucial when aiming for success in software development projects. This aligns with the studies of Chiang and Mokerjee (2004) and Lindstrom and Jeffries (2004) who state the importance of these three components in software development productivity. The question aimed to find out the connection amongst people, principles and technological processes, in order to see how do managers engage their employees to use technology and how does this one help to the productivity increase. When answering this question, the interviewees claim that this is something done spontaneously without proper rules or instructions. However,

they see the human capital as the key, principles as guidelines, and technological processes as a way to ensure that people can follow the principles (L. Zhang, email, June 13, 2013).

The view of Blackburn, *et al*., (1996) who state that special attention should be payed to people because they dominate tools and techniques, is approved by the participants of this study who bring up examples of what their respective organizations do to support their human capital. One of the participants states that managements tries to be supportive and play a ”servant-leader” role in an upside-down organization in order to facilitate people’s work at this point (J. Hermann, email, July 2, 2013). They engage people by training them, showing demos and providing occasions to practice, in order to make them familiar with new concepts and technologies. This benefits their working processes as the technological solutions are considered important to improve software development speed and reduce the communication overhead (Chiang and Mokerje, 2004).

After that, managers/product owners communicate to the team what the customer wants, by clearly determining the requirements. According to the participants, this is mostly done by creating an appropriate product backlog from which the team should pick work for the iterations. These practices align with Cordeiro’s, *et al.* (2008) and Oversen’s (2012) descriptions of the product backlog and its importance to the software development processes.

In some organizations, every time that the team efforts produce successful results, management rewards the team members by organizing some ”free-time” activities such as going out for lunch/beer, watch T.E.D videos about certain topics that the team finds interesting, etc. (V. Thavonekham, M. Gustafsson, M. Jonsson, personal interviews, June-July, 2013). These are important for the team motivation because, as V. Thavonekham (personal interview, June 27, 2013) states, working with software development projects is tough and sometimes turns out to be stressful as it often creates the feeling of a long process that never ends. Since Scrum has changed this mindset introducing the ”sprint” concept, people have to stop time after time and experience some satisfaction about meeting certain milestones (M. Jonsson, personal interview, June 25, 2013). This is something that mostly happens after a sprint is finished successfully, so these small entertaining activities are known as “sprint celebrations”. The fact that organizations organize such free time activities confirms previous researchers’ work, which states that management should support and motivate their teams (Megerian and Sosik, 1997; Boddy

and David, 2001; Maylor, 2010); try to deal with their emotions (Humphrey, 2002) and provide various incentives (Milne, 2007; American Economic Association, 2008), in order to help them deal with feelings of frustration/optimism and achieve high job performance (Gentry and Weber, 2007).

4.8. Management actions that affect productivity

The majority of the participants, when trying to find out how do managers affect the productivity of their teams, prefer to think of project management in an agile world.

One of the participants points out the difference between macro and micro management styles, and aligns the agile methodologies with the macro one as they are more rapid and give developers freedom to think and act, allowing them to be creative. This is the reason why everyday Scrum masters organize stand-up meetings to specify the requirements to the developers and let them free to accomplish their daily tasks according to their own visions. This observation is supported by the theoretical framework of Ewing (1990), Huselid and Becker (2011) and Priem, *et al.*, (2011) who explain macro and micro management styles and point out the advantages of executing a macro management style.

The daily meetings, as the interviewees claim, are also important to help the developers create the agile state of mind, relieve stress and perceive to what extent they have to be responsible for their individual tasks, always considering the fact that they are part of a whole team. All the participants consider important the autonomy of the individuals involved in software development projects. Attitude is important, too. Instead of controlling and directing, managers should try to support their teams and ensure that they have a steady and smooth supply of work. This aligns with Megerian's and Sosik's (1997), Boddy's (2001) and Maylor's (2010) theories about influencing and motivating the teams, leading them towards meeting and exceeding the expected outcomes. The majority of the interviewees state that for the team motivation, it is vital to allow the teams to be self-organizing by providing them all they need. Indeed, this is proved by Lindstrom and Jeffries (2004) who assert that the self-managing teams can be effective since the decision-making process is executed by competent individuals who are close to the project details, aware of the project goals and supported by top management.

Making sure that the right people are carrying the right tasks and ensuring communication among the members, by helping them to follow their own process, is a management

strategy that has resulted successful so far. The communication and more specifically, the direct contact with the customer is another management action that would lead to the productivity improvement of the team.

> … The teams should be also allowed to have customer contact directly, not by a proxy management and other similar ways. Sometimes, the product owner draws up the big sketch, the initial ideas on what the customer wants, and after that, it is the team who interacts with the customer. On my opinion, it is more effective than having a part in between the customer and the person who is working on the feature, because otherwise it becomes like the whispering game: things get distorted along the way (M. Jonsson, personal interview, June 25, 2013)

The importance of establishing direct contact between the customer and the development team is confirmed by Bosch's and Bosch-Sijtsema's (2011) studies that affirm that it affects engagement and motivation of the development team, which on the other side are directly connected with the productivity of the team. However, as one of the participants claims, this does not seem to be easy, especially in those large-sized projects, that are compounded by many, many teams and that have several levels of product owners (S. Singman, personal interview, June 17, 2013)

Summarizing the results of the interviews and questionnaires, four out of eleven interviewees hold more important the quality than the speed of the development processes. Nevertheless, all the participants recognize some factors that may hamper or enhance software development speed (Table 4.1).

Management actions that support productivity
• Minimizing the interruptions of software-developer mental flow • Removing the impediments of the development team • Helping the staff to understand the methodology applied • Setting up clear processes, structures, etc. to ensure that members of the same team will work the same way • Empowering the development team to make decisions, take actions and learn from their own mistakes • Ensuring that the quality is built from the very beginning • Collaborating closely with the testers from the early stages of a project • Managing the project risk with the intention to reduce and minimize it
Management actions that hamper productivity

- A micro-management style where you schedule the work of the developers and prioritize their tasks
- The lack of incentives (moral, monetary, etc)
- Interrupting the work of the developers (sprint interruption, a design stop, etc.)
- Changing the productivity of the tasks when the work has already started
- Scheduling additional unplanned meetings and involving them in a lot of activities that were not planned initially.
- Assigning additional scheduled work to the developers instead of letting them be autonomous and self-organizing.
- Skipping tasks without completing them, aiming for the speed more than for the quality, but falling continuously in re-work loops
- The dependency on external parties
- A lack of agreements, instructive procedures etc. to lead developers to the right tasks instead of letting them make assumptions.
- Not defining clearly the acceptance criteria

Table 4.1. Management actions that affect productivity

The above table is created based on the experience of J. Hermann (email, July 2, 2013) and NK Shrivastava (personal interview, June 23, 2013), whereas T. Vermeerschen (email, July 3 , 2013) provides a more detailed explanation and concrete examples.

Most of these points of view which are based on the working experience of the participants are supported by the studies used to create the theoretical framework of this thesis. For example, the fact that the lack of incentives can affect negatively the performance of an organizations by decreasing the morale of its employees, was studied initially by Milne (2007) and the American Economic Association (2008). However it is interesting to observe how none of the participants placed importance on the monetary incentives. It seems like all their teams are looking forward mainly to “free time activities” and other moral incentives instead of pretending to be awarded a monetary price for accomplishing successfully their tasks. Another factor to be mentioned is interrupting the work of the developers. Schwaber and Sutherland (2011) state that sprint cancellations can increase stress, create the feeling of frustration leading to a productivity decrease of the developers.

Whatever the management actions, it is important to keep into consideration the team morale. Productivity and speed do not always go in parallel with the team morale, the

motivation. Managers have to behave in the right way to create a comfortable working environment for their teams, to influence and motivate them, while avoiding to create them feelings of indifference and frustration. The experience of the participants confirms the previous studies that software development team should be emotionally attached to the project's goals and milestones, otherwise they will not put extra efforts, nor will they try their best.

To sum up, managing teams is not easy, but it is crucial to the success of their work, so managers must be aware of the consequences of their managing actions.

4.9. Suggestions based on the participants' experience

After having created the literature framework and after having seen the factors that may lead the software development project to failure or success, and the importance of agile and lean approaches for the development team, the author has developed two additional questions:

- **Which practices are considered the best to be implemented when developing software?**
- **What would you suggest organizations to do in order to improve productivity and increase business value in software development industry?**

4.9.1. Best practices to develop software

When asked about the practices that are considered the best to be implemented when developing software, the interviewees come up with a variety of good practices, among which the most important are:

- Determining clearly the requirements and having a good understanding of them before starting to write code
- Testing faster
- Create joint reviews so everybody (developers, business analysts, etc.) have a common understanding
- Allowing the teams to be self-organizing rather than controlling and scheduling their work
- Encouraging and supporting communication and discussion among the team members

- Creating agreements and clear processes for everybody
- Iterations are important – the same iteration length for all the teams to ensure effective cooperation.
- Creating coherent teams –avoiding to change the team members during the projects.
- Ensuring tight feedback loops among all the team members;
- Time-boxing
- Limiting work in progress (WIP).
- Knowing what it means and how to be on track
- Aiming for continuous improvement
- Adapting to the context
- Getting rapid feedback
- Giving the customer early opportunity to accept delivery
- Knowledge sharing and transfer
- Getting the right skills engaged in the work
- Making sure that technical debt doesn't build up
- Promoting the sprint success

4.9.2. Suggestions on productivity improvement

Many years of experience in software development industry have helped the participants of this study create a certain feeling about “DO”-s and “DON’T”-s when it to productivity improvement. In an overall observation, when being asked this question, all the interviewees reflect on situations when things went wrong and try to express the lessons learned in form of advices. Among the suggested ones are:

- Measure productivity, find the root causes of possible problems and fix them
- Eliminate useless work activities in favor of high-value activities
- Assign the right tasks to the competent people, in other words letting everybody work on what they are best on
- Motivate the team members to commit on what they are doing
- Ensure customer contact for the team members
- Use the agile to improve the team working efficiency and improve the team velocity continuously.
- Involve testing soon and make it as automatic as possible.

- Focus on training people and building competence. Help people to learn and improve.
- Preserve the existing procedures and agreements if they proved to be useful in the past. “Agile” does not mean “no procedures or processes”.
- Try to make things right the first time. Define as good stories those that contain enough correct information, and prioritize them.
- Focus on improving the productivity of the teams rather than that of specific individuals.
- Organize small activities that make the team happy (sprint celebrations)
- Consider change management, ask for feedback and try to be opened

The respondents of the online questionnaires, after being asked the same question, came up with the following suggestions:

- Criticize and combine the development methods.
- Make sure that the development team understands the business value of what is delivered.
- Apply lean startup techniques to understand what the customer really wants and shorten the feedback as much as possible.
- Include User Experience in the evaluation of how well a product is doing and in evaluating what is produced.
- Do not assume that everyone is going to be motivated by the same free activity or piece of free cake! Encourage knowledge sharing and having actual cross-functional teams.
- Time based assignments with some target benefit attached to performance result will increase both the developers’ productivity and business value.

It is interesting to see how one of the participants relates productivity to the management style, thinking that the problem that the real issue of the organizations nowadays are those managers that think they know better.

> It is this shortage of understanding by Management that I would say is the single biggest impediment to improvement in the industry as a whole, because it has everybody pulling in different directions. A thoughtless few minutes work by a manager can undo weeks of hard work toward an improvement by a team.
>
> An example? I recall setting up a process improvement process for one organization. The managers liked the idea and took over the project. The resultant

> process was designed to manage single improvements through a series of stage gates that were tied to the financing of the work. It was a process totally unsuited to managing process improvements because it completely ignored the engineering process that would be necessary to ensure a change would be an improvement; it completely ignored the fact that improvements were a "many-to-many" relationship with the inputs to the process, and it completely ignored the reality that to get an improvement to stick you need to involve and get buy-in at several stages in the process from the people who would be working in a different way as a result of the change. Luckily, it also completely ignored that many changes need very little finance, so we just made a series of small changes and ignored their process. The managers saw improvement as something designed by experts and imposed on the team (as you might if designing a production process for low-skilled workers, maybe) (P. Oldfield, email, June 28 , 2013).

All the participants see the human factors as the key to motivate and engage employees. All of them stress the importance that should be placed to studying the human behavior and being emotionally intelligent for the benefit of the team work, for the achievement of high results that lead to increased business value. This aligns with the Blackburn's *et al*., (1996) and later on with George's (2000), Boddy's and David's (2001), Kerr's, *et al.* (2006) and Maylor's (2010) studies that state that empoyees should be motivated and influenced to achieve higher results. Being emotionally intelligent with the other team members increases employee satisfaction and job performance (Gentry and Weber, 2007).

The author of this study observes a tendency of some interviewees to focus more on the technical aspects of managing the work of the development teams then on the "soft issues" of management, even though they retain them to be important. Probably, it is due to their educational and/or professional background that they tend to shift the attention from the project management practices to the engineering ones, often relating productivity or business value to practical issues, such as time – boxing, iterations, automated testing, big bang theory, etc. that were not the focus of this study. They suggest ways to move a development team to agile, but other than suggestions they seem to not have any appropriate fixed procedures to accomplish it. The interviewees tend to find a connection between productivity and job satisfaction. However, when it is to the benefits that the use of lean and agile methodologies brings to the business, they tend to see the impacts of these methodologies on the working processes more than on the team themselves. On the author's opinion it might be due to the nature of software development domain that often tends to evaluate the technical/engineering skills of the developers and the automation of the working processes, putting less emphasis on people's skills and emotions, and

consequently on the soft aspects of management that might enhance the productivity of the team.

5. CONCLUSIONS

The main aim of this study was to observe how an integration of agile and lean approaches into the software development project management processes would enhance productivity and would increase the business value of respective organizations. This chapter summarizes the main findings of this study, acknowledges the limitations of it and provides suggestions for further research.

5.1. Summary of the main findings

The literature review has shown that software development industry is a volatile domain, which continuously calls for the need of in-time delivery of the required products. On the other side, these products should meet the user needs and fulfill the quality standards. The ability to deal with the unpredictable customer requirements is ensured by integrating agile approaches in software development processes. The agile methodologies allow the customer to provide continuous feedback, eliminating the risk that the team might deliver a product that does not fulfill the customer needs anymore by the time that it reaches the market. The other advantages associated with agile approaches such as small to medium collocated development teams, the autonomy to organize and make decisions about their work, etc. improve communication among team members and increase their productivity.

Indeed, the productivity of the team is a main factor to be considered when aiming to achieve organizational success, therefore the aim of this research was to find out ways to measure an improve productivity. Apparently, what practice shows is that among the most common metrics that the organizations use nowadays are function-points productivity, story-points productivity, customer satisfaction, etc. The interviewees suggest avoiding to create very big development teams as this might result in communication overhead. This often translates in slower working procedures and higher costs due to the productivity decrease. On the other side, they claim that there is no point in organizing meetings and scheduling other activities when the team is compounded by 2-3 developers as this would be time consuming and unnecessary. Therefore, the ideal size, as the previous scholars' research claims and as the current findings support, is between 5-9 developers per team. Literature shows that teams should organize work in development cycles in order to release product time after time and allow the customer to provide feedback. According to the participants of this study, the suggested length of these development cycles is between 2-5

weeks, with a strong emphasis on the three-week cycle as it has been observed to be the most effective one so far.

According to the literature and to the current findings, agile methodologies are seen to be crucial to the success of the software development organizations due to the benefits associated with them. Consequently, the interviewed Scrum masters claim that in those projects that have implemented agile, it is crucial for all the members of the developing team to be knowledgeable of agile principles, no matter their specific role in the project.

Due to the fact that operating agile, according to previous researchers, means embedding a totally new mindset into the organizational culture, this process is often associated with a lot of challenges. Perhaps the most important one that the participants of this study claim to have faced is knowledge sharing and transfer. They align it with some developers' attitude to work individually and be micro-managed, instead of collaborating in self-organizing teams. Other challenges, that Scrum masters have faced so far, are the coordination among dispersed teams in large-sized projects, the establishment of direct contact with the customer, the tendency of the developers to focus on specific tasks instead of taking a "helicopter view" and trying to understand the whole picture of the product that the customer wants, etc. Nevertheless, experience has discovered many ways to deal with these challenges. Perhaps the most common ones are coaching the team, providing examples of organizations that have succeeded in implementing agile, implementing it in a small segment of the organization and afterwards in the whole one, etc.

Despite the benefits of agile, several scholars claim that it still cannot be used to eliminate waste. According to their research, eliminating waste and increasing business value are typical features of the lean production philosophy. Therefore, a need to explore the lean software development field comes to the interest of the reader.

As revealed in literature, lean in software development helps to reduce bugs, waiting time, extra features and processes, task switching, unfinished work, etc. leading to quality software developed in a short time and with low cost. Yet, as various scholars argue, it cannot help in meeting the changeable customer requirements. Therefore, a combination of both leanness and agility would be the ideal approach to ensure organizations high economical profits.

The theoretical framework recognizes the leagility paradigm, as a model that can realize the combination of both agile and lean methodologies by using a decoupling point as a separator. On the upstream side of the decoupling point lay the lean processes, and the agile ones lay on the downstream side. This study investigates that some organizations that have tried to explore and implement this paradigm. From the interviews and questionnaires, one can draw the conclusion that despite its perceived benefits, it remains a new concept to the software development industry. It results that today organizations have tried to embed both agile and lean principles into their developing processes, but for a number of reasons, they have not find it reasonable or possible to do it within the same team.

Literature also demonstrates that methodologies and technology are vital to achieve successfully the positive outcomes from software developing projects, but people are the first and most important factor to be considered. The authors of the papers included in this study assert that management focus is shifting from finding the best working practices, to finding the best management practices to influence and motivate the team members to engage on what they do. The participants of this study identify a number of management actions that would hamper software development speed and productivity, such as not clarifying the acceptance criteria, interrupting the developers' work, micro managing the teams, etc. The interviewees declare that these and other similar factors would lead employees to routine work, which kills their creativity and reduces the productivity at work. Instead, they suggest that managers should be supportive and emotionally intelligent, trying to understand the needs of the team, ensure them sufficient contact with the customer, organize stress-relief activities, etc.

The participants of this study suggest several ways to improve productivity such as: encouraging customer contact, involving testing as soon as possible, training staff and building competence, eliminating useless activities, working on embracing change management, etc.

However, the author observes a tendency of some participants to focus more on practical details than on soft issues of management when talking about the effects of embedding agile and lean approaches in development processes. The reason why they perceive the productivity of the team as something related more to the external factors than to the developers' wellbeing and inner emotions about certain working practices, might be either

due to their engineering background, or due to the nature of the industry in which they operate.

5.2. Limitation and suggestions for further research

As mentioned in the first chapter, this study suffers from several limitations such as the few number of participants involved, the fact that all the interviewees are Scrum masters, etc. It would be interesting to conduct a larger study where not only Scrum masters, but also Scrum practitioners would be involved. It would also be interesting to work closely with a development team in order to investigate how they integrate agile and lean principles into their working practices. Additional potential aspects to look into would be firstly finding out what does a development team consider important for their productivity and afterwards observing how would each of those factors impact the outcomes of their work.

REFERENCES

Agarwal, A., Shankar, R. and Tiwari, M. (2006) 'Modeling the metrics of lean, agile and leagile supply chain: An ANP-based approach', *European Journal of Operational Research*, 173(1), pp. 211–225 ScienceDirect [Online]. Available at: http://www.sciencedirect.com/science/article/pii/S0377221705000135 (Accessed: 14 May 2013)

Ambler, S. (2009) Scaling Agile Software Development Through Lean Governance, *ICSE Workshop on Software Development Governance,* Vancouver, BC 17-17 May 2009 . IEEE Available at: http://xplqa30.ieee.org/xpl/articleDetails.jsp?tp=&arnumber=5071328&contentType=Conference+Publications (Accessed: 3 February 2013).

American Economic Association (2008) 'Intrinsic Motivation and Incentives', *The American Economic Review*, 98(2), pp. 201-205 AEA [Online]. Available at: https://www.aeaweb.org/articles.php?doi=10.1257/aer.98.2.201 (Accessed: 23 February 2013).

Ballé, F. and Ballé, M. (2005) *Lean development*, London: London Business School .

Banomyong, R., Veerakachen, V. and Supatn, N. (2008) 'Implementing leagility in reverse logistics channels'. *International Journal of Logistics Research and Applications: A Leading Journal of Supply Chain Management*, 11(1), pp. 31-47 Taylor Francis Online [Online] Available at: http://www.tandfonline.com/doi/abs/10.1080/13675560701403651#.UjHGwdKviSo (Accessed: 30 April 2013).

Beck, K., Grenning, J., Martin, R., Beedle, M., Highsmith, J., Mellor, S., Bennekum, A. V., Hunt, A., Schwaber, K., Cockburn, A., Jeffries, R., Sutherland, J., Cunningham, W., Kern, J., Thomas, D., Fowler, M., Marick, B. (2001) '*Manifesto for Agile Software Development'*. [Online] Available at: http://www.agilemanifesto.org/ (Accessed: 8 May 2013)

Blackburn, J. D., Scudder, G. D. and Van Wassenhove, L. N. (1996) 'Improving speed and productivity of software development: a global survey of software developers'. *The IEEE Transactions on Software Engineering*, 22(12), pp. 875 - 885 IEEE [Online]. Available at:

http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=553636 (Accessed: 7 January 2013)

Boddy, D. (2001) '*Managing Projects: Building and Leading the Team'*. Essex: Financial Times/ Prentice Hall.

Bosch, J. and Bosch-Sijtsema, P. M. (2011) 'Introducing agile customer-centered development in a legacy software product line'. *Software: Practice and Experience*, 41(8), pp. 871–882.

Bosch, J. and Bosch-Sijtsema, P. (2010) 'Coordination Between Global Agile Teams: From Process to Architecture', In: D. Šmite, N. B. Moe and P. J. Ågerfalk, (ed.) *Agility Across Time and Space: Implementing Agile Methods in Global Software Projects*. Berlin: Springer Berlin Heidelberg, pp. 217-233SpringerLink [Online]. Available at: http://link.springer.com/chapter/10.1007%2F978-3-642-12442-6_15 (Accessed: 3 March 2013)

Braun, E. (2005) 'Lean/agile Methods for Web Site Development', *Online*, 29(5), pp. 58-60.

Cachia, M. and Millward, L. (2011) 'The telephone medium and semi-structured interviews: a complementary fit', *Qualitative Research in Organizations and Management: An International Journal*, 6(3), pp. 265 – 277 Emerald [Online]. Available at: http://www.emeraldinsight.com/case_studies.htm/case_studies.htm?articleid=17003597&show=html (Accessed: 23 December 2012)

Carruthers, J., (1990) 'A Rationale for the Use of Semi-structured Interviews'. *Journal of Educational Administration*, 28(1), pp. 63-68 Trove [Online]. Available at: http://trove.nla.gov.au/work/153182499?q&versionId=166940203 (Accessed: 23 December 2012)

Cawley, O., Wang, X. and Richardson, I. (2010) 'Lean / agile software development methodologies in regulated environments – state of the art'. In: S. B. Heidelberg, (ed.) *Lecture Notes in Business Information Processing.* Helsinki: Springer Berlin Heidelberg, pp. 31-36 SpringerLink [Online]. Available at: http://link.springer.com/chapter/10.1007%2F978-3-642-16416-3_4 (Accessed: 18 March 2013)

Chau, T., Maurer, F. and Melnik, G. (2003) 'Knowledge Sharing: Agile Methods vs. Tayloristic Methods'. *WETICE '03 Proceedings of the Twelfth International Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises*, Linz, Austria. 9-11 June 2003. IEEE Computer Society. Available at: http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=1231427 (Accessed: 13 June 2013)

Chiang, I. R. and Mokerjee, V. S. (2004) 'Improving software team productivity', *Communications of the ACM - New architectures for financial services*, 47(5), pp. 89-93 ACM [Online]. Available at: http://dl.acm.org/citation.cfm?id=986217 (Accessed: 28 January 2013)

Collofello, J. S., Woodfield, S. N. and Gibbs, N. E., (1983) 'Software productivity measurement', in *AFIPS National Computer Conference*, AFIPS Press, pp. 757-762

Conti, R. et al. (2006) 'The effects of lean production on worker job stress', *International Journal of Operations and Production Management*, 26(9), pp. 1013 – 1038 Emerald [Online]. Available at: http://www.emeraldinsight.com/case_studies.htm/case_studies.htm?articleid=1567131&show=html&PHPSESSID=t54hrh6bv8tbl61ctr1hohrqk1 (Accessed: 17 April 2013)

Cordeiro, L., Mar, C., Valentin, E., Cruz, F., Patrick, D., Barreto, R., Lucena, V. (2008) 'An Agile Development Methodology Applied to Embedded Control Software under Stringent Hardware Constraints', *Software Engineering Notes*, 33(1), pp. 1-10 ACM [Online]. Available at: http://dl.acm.org/citation.cfm?id=1344459 (Accessed: 17 April 2013)

Crandall, S. J. and McGaghie, W. C. (2001) 'Discussion and Conclusion: Interpretation', *Academic Medicine*, 76(9), pp. 942-944 AAMC [Online]. Available at: http://journals.lww.com/academicmedicine/fulltext/2001/09000/discussion_and_conclusion__interpretation.30.aspx (Accessed: 16 December 2012)

Dean, J. W. and Bowen, D. E., (1994) 'Management Theory and Total Quality: Improving Research and Practice through Theory Development', *The Academy of Management Review*, 19(3), pp. 392-418 EBSCO Host Connection [Online]. Available at: http://connection.ebscohost.com/c/articles/9412271803/management-theory-total-quality-improving-research-practice-through-theory-development (Accessed: 13 April 2012)

Dinakar, K., (2009) ‘Agile development: overcoming a short-term focus in implementing best practices’, *OOPSLA '09 Proceedings of the 24th ACM SIGPLAN conference companion on Object oriented programming systems languages and applications*, New York, NY, USA 25 – 29October 2009. ACM. Available at: http://dl.acm.org/citation.cfm?id=1639950.1639952 (Accessed: 15 November 2012)

Dingsøyr, T., Nerur, S., Balijepally, V. and Moe, N. B. (2012) ‘A decade of agile methodologies: Towards explaining agile software development’, *The Journal of Systems and Software*, 85(6), pp. 1213– 1221 ScienceDirect [Online]. Available at: http://www.sciencedirect.com/science/article/pii/S0164121212000532 (Accessed: 16 November 2012)

Dong, W. Y., (1995) ‘Lean production and industrial engineering applied in China’, *Computers and Industrial Engineering*, 29(1-4), pp. 233-237 Emerald [Online]. Available at: http://www.emeraldinsight.com/journals.htm?issn=2040-4166&volume=1&issue=3&articleid=1876544&show=ref&PHPSESSID=mfa7jkm3jspp2scnrvtmchlaa3 (Accessed: 12 June 2013)

Duffy, J. R. (2005) ‘Critically appraising quantitative research’. *Nursing and Health Sciences*, 7(4), pp. 281–283 PublMed [Online]. Available at: http://www.ncbi.nlm.nih.gov/pubmed/16271135 (Accessed: 7 January 2013)

Duncan, A. (1988) ‘Software development productivity tools and metrics’, ICSE '88 *Proceedings of the 10th International Conference on Software Engineering*, Los Alamitos, CA, USA, 11-15 Apr 1988. ACM. Available at: http://dl.acm.org/citation.cfm?id=55828 (Accessed: 15 November 2012)

Dyba, T. and Dingsoyr, T. (2009) ‘What Do We Know about Agile Software Development?’, *Software IEEE*, 26(5), pp. 6-9 IEEE [Online]. Available at: http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=5222784&url=http%3A%2F%2Fieeexplore.ieee.org%2Fxpls%2Fabs_all.jsp%3Farnumber%3D5222784 (Accessed: 16 November 2012)

Eaton, M. (2003) ‘Running the leagile marathon - lean/agile improvement programmes’, *Manufacturing Engineer*, 82(6), pp. 14-17 IEEE [Online]. Available at: http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=1267740&url=http%3A%2F%2Fie

eexplore.ieee.org%2Fxpls%2Fabs_all.jsp%3Farnumber%3D1267740 (Accessed: 16 November 2012)

Emiliani, M. (1998) 'Lean behaviors', *Management Decision,* 36(8), pp. 615 – 631 Emerald [Online]. Available at: http://www.emeraldinsight.com/journals.htm?articleid=865039 (Accessed: 14 April 2013)

Ewing, R. P. (1990) 'Moving from micro to macro issues management', *Public Relations Review*, 16(1), pp. 19–24 ScienceDirect [Online]. Available at: http://www.sciencedirect.com/science/article/pii/S0363811105800335 (Accessed: 28 June 2013)

Fogelström, N. D., Gorschek, T., Svahnberg, M. and Olsson, P. (2010) 'The impact of agile principles on market-driven software product development', *Journal of Software Maintenance and Evolution: Research and Practice,* 22(1), pp. 53–80 Wiley Online Library [Online]. Available at: http://onlinelibrary.wiley.com/doi/10.1002/spip.420/abstract (Accessed: 16 November 2012)

Geller, E. S. (2000) *Empathic Leadership.* Virginia: Safety Performance Solutions (SPS) '

Gelo, O., Braakmann, D. and Benetka, G. (2008) 'Quantitative and Qualitative Research: Beyond the Debate', *Integrative Psychological and Behavioral Science,* 42(3), pp. 266-290 SpringerLink [Online]. Available at: http://link.springer.com/article/10.1007%2Fs12124-008-9078-3 (Accessed: 7 January 2013)

Gentry, W. A. and Weber, T. J. (2007) *Empathy in the workplace: A tool for effective leadership.* Center for Creative Leadership [Online]. Available at: http://www.ccl.org/leadership/pdf/research/EmpathyInTheWorkplace.pdf (Accessed: 28 January 2012)

George, J. M. (2000) 'Emotions and Leadership: The Role of Emotional *Intelligence', Human relations,* 53(8), pp. 1027-1055 SAGE Journals [Online]. Available at: http://hum.sagepub.com/content/53/8/1027.abstract (Accessed: 18 March 2012)

Germain, É. and Robillard, P. N. (2005) 'Engineering-based processes and agile methodologies for software development: a comparative case study'. *Journal of Systems and Software - Special issue: Software engineering education and training,* 75(1-2), pp.

17-27 ACM [Online]. Available at: http://dl.acm.org/citation.cfm?id=1063023 (Accessed: 16 November 2012)

Goldsby, T. J., Griffis, S. E. and Roath, A. S. (2006) 'Modeling Lean, Agile and Leagile supply chain strategies'. *Journal of Business Logistics*, 27(1), pp. 57-80. Wiley Online Library [Online]. Available at: http://onlinelibrary.wiley.com/doi/10.1002/j.2158-1592.2006.tb00241.x/abstract (Accessed: 5 June 2013)

Greer, D. and Hamon, Y. (2011) 'Agile Software Development', *Software – Practice and Experience*, 41(9), pp. 943–944 Wiley Online Library [Online]. Available at: http://onlinelibrary.wiley.com/doi/10.1002/spe.1100/abstract (Accessed: 12 March 2013)

Groves, K. S. (2006) 'Leader emotional expressivity, visionary leadership, and organizational change', *Leadership and Organization Development Journal*, 27(7), pp. 566 – 583 Emerald [Online]. Available at: http://www.emeraldinsight.com/journals.htm?articleid=1572834 (Accessed: 22 March 2012)

Gugiu, P. C. and Rodríguez-Campos, L. (2007) 'Semi-structured interview protocol for constructing logic models', *Evaluation and Program Planning*, 30(4), pp. 339-350 PublMed [Online]. Available at: http://www.ncbi.nlm.nih.gov/pubmed/17905433 (Accessed: 7 January 2013)

Gunasekaran, A. (1999) 'Agile manufacturing: A framework for research and development', *International Journal of Production Economics*, 62(1-2), pp. 87–105 ScienceDirect [Online]. Available at: http://www.sciencedirect.com/science/article/pii/S0925527398002229 (Accessed: 12 March 2013)

Hart, C. (2012) *Doing your masters dissertation*. London: Sage Publications Ltd.

Hernon, P. and Schwartz, C., (2009) 'Procedures: Research design (Editorial)', *Library and Information Science Research*, Volume 31, pp. 1-2.

Highsmith, J. and Cockburn, A. (2001) 'Agile software development: the business of innovation'. *Computer*, 34(9), pp. 120 – 127 IEEE [Online]. Available at: http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=947100&url=http%3A%2F%2Fiee

explore.ieee.org%2Fxpls%2Fabs_all.jsp%3Farnumber%3D947100 (Accessed: 26 November 2012)

Highsmith, J. (2002) *Agile software development ecosystems.* Boston, MA: Addison-Wesley Longman Publishing Co., Inc..

Humphrey, R. H. (2002) 'The Many Faces of Emotional Leadership', *The Leadership Quarterly*, 13(5), pp. 493-504 IngentaConnect [Online]. Available at: http://www.ingentaconnect.com/content/els/10489843/2002/00000013/00000005/art00140 (Accessed: 19 March 2012)

Huselid, M. A. and Becker, B. E. (2011) 'Bridging Micro and Macro Domains: Workforce Differentiation and Strategic Human Resource Management', *Journal of Management*, 37(2), pp. 421-428 SAGE Journals [Online]. Available at: http://jom.sagepub.com/content/37/2/421.abstract (Accessed: 29 April 2013)

Jo, S. and Shimb, S. W. (2005) 'Paradigm shift of employee communication: The effect of management communication on trusting relationships'. *Public Relations Review*, 31(2), pp. 277–280 ResearchGate [Online]. Available at: http://www.researchgate.net/publication/222745493_Paradigm_shift_of_employee_communication_The_effect_of_management_communication_on_trusting_relationships?citationList=incoming (Accessed: 19 March 2012)

Jones, R. and Murray, N. (2008) *Change, strategy and projects at work.* Oxford: Butterworth-Heinnemann.

Keppel, G. and Zedeck, S. (1990) 'Data Analysis for Research Designs: Analysis of Variance and Multiple Regression/Correlation Approaches', *Contemporary Sociology*, 19(3), pp. 486-487 JSTOR [Online]. Available at: http://www.jstor.org/stable/2072532 (Accessed: 5 January 2013)

Kerr, R., Garvin, J., Heaton, N. and Boyle, E. (2006) 'Emotional intelligence and leadership effectiveness. *Leadership and Organization Development Journal*, 27(4), pp. 265 – 279 Emerald [Online]. Available at: http://www.emeraldinsight.com/journals.htm?articleid=1558663 (Accessed: 19 March 2012)

Kircher, M. and Hofman, P. (2012) ‘Combining Systematic Reuse with Agile Development - Experience Report’, *SPLC '12 Proceedings of the 16th International Software Product Line Conference - Volume 1*, Salvador, Brazil 2-7 September 2012. ACM. Available at: http://dl.acm.org/citation.cfm?id=2362536&picked=prox (Accessed: 12 April 2012)

Kreps, D. M. (1997) ‘Intrinsic Motivation and Extrinsic Incentives’, *The American Economic Review*, 87(2), pp. 359-364 JSTOR [Online]. Available at: http://www.jstor.org/discover/10.2307/2950946?uid=2&uid=4&sid=21102570064597 (Accessed: 11 March 2012)

Krishnamurthy, R. and Yauch, C. A. (2007) ‘Leagile manufacturing: a proposed corporate infrastructure’, *International Journal of Operations and Production Management*, 27(6), pp. 588 – 604 Emerald [Online]. Available at: http://www.emeraldinsight.com/journals.htm?articleid=1603053 (Accessed: 3 June 2013)

Liker, J. K. (2004) *The Toyota Way: 14 Management Principles from the World's Greatest Manufacturer.* New York, NY: McGraw-Hill, Inc.

Liker, J. and Meier, D. (2006) *The Toyota Way Fieldbook: A Practical Guide for Implementing Toyota's 4Ps.* New York; London: McGraw-Hill, Inc.

Lindstrom, L. and Jeffries, R. (2004) ‘Extreme programming and agile software development methodologies’, *Information Systems Management*, 21(3), pp. 41-52 Emerald [Online]. Available at: http://www.emeraldinsight.com/bibliographic_databases.htm?id=1321475&show=abstract (Accessed: 16 November 2012)

Livermore, J. A. (2008) ‘Factors that Significantly Impact the Implementation of an Agile Software Development Methodology’, *Journal of Software*, 30(4), pp. 31-36 BibSonomy [Online]. Available at: http://www.bibsonomy.org/bibtex/266abdd8808461bfa28791c5daf002d3c/dblp (Accessed: 16 November 2012)

Lu, W., Olofsson, T. and Stehna, L. (2011) ‘A lean-agile model of homebuilders’ production systems’, *Construction Management and Economics*, 29(1), pp. 25-35 Taylor Francis Online [Online] Available at:

http://www.tandfonline.com/doi/abs/10.1080/01446193.2010.531027#.UjM5ltKviSo (Accessed: 5 June 2013)

Mahoney, J. and Goertz, G. (2006) 'A Tale of Two Cultures: Contrasting Quantitative and Qualitative Research', *Political Analysis,* 142(3), pp. 227-249 Oxford Journals [Online]. Available at: http://pan.oxfordjournals.org/content/14/3/227.short (Accessed: 5 January 2013)

Mason-Jones, R., Naylor, B. and Towill, D. R. (2000) 'Engineering the leagile supply chain', *International Journal of Agile Management Systems,* 2(1), pp. 54 – 61 Emerald [Online]. Available at: http://www.emeraldinsight.com/journals.htm?articleid=852407 (Accessed: 17 February 2013)

Mason-Jones, R., Naylor, B. and Towill, D. (2000) 'Lean, agile or leagile? Matching your supply chain to the marketplace', *International Journal of Production Research,* 38(17), pp. 4061-4070 Taylor Francis Online [Online]. Available at: http://www.tandfonline.com/doi/abs/10.1080/00207540050204920?journalCode=tprs20#.UjM7f9KviSo (Accessed: 17 February 2013)

Mateosian, R. (2003) 'Managing Software Projects', *IEEE Micro*, 23(4), pp. 11-13 ResearchGate [Online]. Available at: http://www.researchgate.net/publication/3215321_Managing_software_projects (Accessed: 16 November 2012)

Maurer, F. and Melnik, G. (2006) 'Agile methods: moving towards the mainstream of software industry', *ICSE '06 Proceedings of the 28th international conference on Software engineering*, New York, NY, USA 20-28 May 2006. ACM. Available at: http://dl.acm.org/citation.cfm?id=1134285 (Accessed: 11 November 2012)

Maxwell, K. D. and Forselius, P. (2000) 'Benchmarking software development productivity', *Software, IEEE,* 7(1), pp. 80-88 IEEE [Online]. Available at: http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=820015&url=http%3A%2F%2Fieeexplore.ieee.org%2Fxpls%2Fabs_all.jsp%3Farnumber%3D820015 (Accessed: 16 November 2012)

Maylor, H. (2010) *Project Management*. 4th ed., Essex: Pearson Education Limited.

McBride, T. (2008) The mechanisms of project management of software development. *Journal of Systems and Software,* 81(12), pp. 2386–2395 ScienceDirect [Online]. Available at: http://www.sciencedirect.com/science/article/pii/S0164121208001581 (Accessed: 13 April 2013)

Megerian, L. E. and Sosik, J. J. (1997) 'An Affair of the Heart: Emotional Intelligence and Transformational Leadership'. *Journal of Leadership and Organizational Studies*, 3(3), pp. 31-48 SAGE Journals [Online]. Available at: http://jlo.sagepub.com/content/3/3/31.abstract (Accessed: 24 March 2012)

Middleton, P. and Joyce, D. (2012) 'Lean Software Management: BBC Worldwide Case Study'. *IEEE Transactions on Engineering Management,* 59(1), pp. 20 – 32 IEEE [Online]. Available at: http://jlo.sagepub.com/content/3/3/31.abstract (Accessed: 25 June 2013)

Milne, P. (2007) 'Motivation, incentives and organizational culture'. *Journal of Knowledge Management*, 11(6), pp. 28-38 Emerald [Online]. Available at: http://www.emeraldinsight.com/journals.htm?articleid=1631476 (Accessed: 23 January 2013)

Mishra, A. and Mishra, D. (2011) 'A curriculum for agile software development methodologies', *Software Engineering Notes,* 36(3), pp. 1-2 ACM [Online]. Available at: http://dl.acm.org/citation.cfm?id=1968608 (Accessed: 16 November 2012)

Naim, M. M. and Gosling, J. (2011) 'On leanness, agility and leagile supply chains'. *International Journal of Production Economics*, 131(1), pp. 342-354 EconPapers [Online]. Available at: http://econpapers.repec.org/article/eeeproeco/v_3a131_3ay_3a2011_3ai_3a1_3ap_3a342-354.htm (Accessed: 5 June 2013)

Narang, R. V. (2008) 'Some Issues to Consider in Lean Production'. *ICETET '08. First International Conference on Emerging Trends in Engineering and Technology*, Nagpur, Maharashtra 16-18 July 2008. IEEEXplore. Available at: http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=4580000&url=http%3A%2F%2Fieeexplore.ieee.org%2Fxpls%2Fabs_all.jsp%3Farnumber%3D4580000 (Accessed: 11 May 2013)

Naylor, J., Naim, M. M. and Berry, D. (1999) 'Leagility: Integrating the lean and agile manufacturing paradigms in the total supply chain', *International Journal of Production Economics,* 62 (1-2), pp. 107–118 EconPapers [Online]. Available at: http://econpapers.repec.org/article/eeeproeco/v_3a62_3ay_3a1999_3ai_3a1-2_3ap_3a107-118.htm (Accessed: 5 June 2013)

Nerur, S., Mahapatra, R. and Mangalaraj, G. (2005) 'Challenges of migrating to agile methodologies'. *Communications of the ACM - Adaptive complex enterprises,* 48(5), pp. 72-78 ACM [Online]. Available at: http://dl.acm.org/citation.cfm?id=1060712 (Accessed: 16 November 2012)

Newell, D. J., (1992) 'Intention-to-treat analysis: implications for quantitative and qualitative research'. *International Journal of Epidemiology,* 21(5), pp. 837-841 PublMed [Online]. Available at: http://www.ncbi.nlm.nih.gov/pubmed/1468842 (Accessed: 3 January 2013)

Nord, R., Ozkaya, I. and Sangwan, R. (2012) 'Making Architecture Visible to Improve Flow Management in Lean Software Development'. *Software, IEEE,* 29(5), pp. 33 – 39 IEEE [Online]. Available at: http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=6226344&url=http%3A%2F%2Fieeexplore.ieee.org%2Fxpls%2Fabs_all.jsp%3Farnumber%3D6226344 (Accessed: 22 March 2013)

Northover, M., Northover, A., Gruner, S., Kourie, D. G., Boake, A. (2007) 'Agile software development: a contemporary philosophical perspective'. *SAICSIT '07: Proceedings of the 2007 annual research conference of the South African institute of computer scientists and information technologists on IT research in developing countries*, New York, NY, USA 2-3 October 2007. ACM. Available at: http://dl.acm.org/citation.cfm?doid=1292491.1292504 (Accessed: 12 November 2012)

Ovesen, N. (2012) *'Challenges of Becoming Agile - Implementing Scrum in Integrated Product Development',* PhD thesis. Aalborg Universitet, Denmark [Online] Available at: http://www.theinventivestep.net/Ovesen-2012_The-Challenges-of-Becoming-Agile_WEB.pdf (Accessed: 11 February 2013)

Peter, K. and Lanza, G. (2011) 'Company-specific quantitative evaluation of lean production methods'. *Production Engineering,* 5(1), pp. 81-87 SpringerLink [Online].

Available at: http://link.springer.com/content/pdf/10.1007%2Fs11740-010-0276-8.pdf (Accessed: 3 January 2013)

Petersen, K. and Wohlin, C. (2008) 'Software process improvement through the Lean Measurement (SPI-LEAM) 'method', *Journal of Systems and Software*, 83(7), pp. 1275–1287 ACM [Online]. Available at: http://dl.acm.org/citation.cfm?id=1805539 (Accessed: 22 March 2013)

Petersen, K. and Wohlin, C. (2011) 'Measuring the flow in lean software development'. *Software—Practice and Experience*, 41(9), pp. 975-996 Wiley Online Library [Online]. Available at: http://onlinelibrary.wiley.com/doi/10.1002/spe.975/abstract (Accessed: 22 March 2013)

Pil, F. K. and Fujimoto, T. (2007) 'Lean and reflective production: the dynamic nature of production models'. *International Journal of Production Research*, 45(16), pp. 3741-3761 Taylor Francis Online [Online]. Available at: http://www.tandfonline.com/doi/abs/10.1080/00207540701223659#.UekUp9IwdEI (Accessed: 22 March 2013)

Polit, D. F. and Beck, C. T. (2010) "Generalization in quantitative and qualitative research: Myths and strategies'. *International Journal of Nursing Studies*, 47(11), pp. 1451–1458 PublMed [Online]. Available at: http://www.ncbi.nlm.nih.gov/pubmed/20598692 (Accessed: 3 January 2013)

Poppendieck, M. (2007) 'Lean Software Development'. *ICSE COMPANION '07 Companion to the proceedings of the 29th International Conference on Software Engineering*, Washington, DC, USA 20-26 May 2007. IEEE Computer Society. Available at: http://dl.acm.org/citation.cfm?id=1248986 (Accessed: 2 March 2013)

Poppendieck, M. and Cusumano, M. (2012) 'Lean Software Development: A Tutorial. *Software IEEE* , 26(5), pp. 26 – 32 IEEE [Online]. Available at: http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=6226341 (Accessed: 20 March 2013)

Powell, S. S. (2008) 'Writing assignments: effective research skills', *British Journal of Healthcare Assistants*, 2(8), pp. 407 – 410 InterNurse [Online]. Available at:

http://www.internurse.com/cgi-bin/go.pl/library/abstract.html?uid=30865 (Accessed: 3 January 2013)

Power, K. (2010) 'Stakeholder Theory and Agile Software Development'. *PROFES '10 Proceedings of the 11th International Conference on Product Focused Software.* Limerick, Ireland 21 - 23 June 2010. ACM. Available at: http://dl.acm.org/citation.cfm?id=1961258 (Accessed: 23 November 2012)

Priem, R. L., Walters, B. A. and Li, S. (2011) 'Decisions, Decisions! How Judgment Policy Studies Can Integrate Macro and Micro Domains in Management Research'. *Journal of Management*, 37(2), pp. 553-580 SAGE Journals [Online]. Available at: http://jom.sagepub.com/content/early/2010/06/04/0149206310372258.abstract (Accessed: 27 June 2013)

Regehr, G. (2001) 'Presentation of Results', *Academic*, 7(9), pp. 940-942 AAMC [Online]. Available at: http://journals.lww.com/academicmedicine/Fulltext/2001/09000/Presentation_of_Results.29.aspx (Accessed: 3 January 2013)

Rico, D. F. (2011) '*The Business Value of Using Agile Project Management for New Products and Services'.* Available at: http://www.projectmanagement.com/articles/267471/Agile-Business-Value (Accessed: 23 February 2013)

Riezebos, J. and Klingenberg, W. (2009) 'Advancing lean manufacturing, the role of IT', *Computers in Industry*, 60(4), pp. 235–236 ACM [Online] Available at: http://dl.acm.org/citation.cfm?id=1523705 (Accessed: 17 May 2013)

Riggio, R. E. and Lee, J. (2007) 'Emotional and interpersonal competencies and leader development', *Human Resource Management Review*, 17 (4), pp. 418–426 EBSCO Host Connection [Online]. Available at: http://connection.ebscohost.com/c/articles/27531719/emotional-interpersonal-competencies-leader-development (Accessed: 12 March 2012)

Ruparelia, N. B. (2010) 'Software Development Lifecycle Models'. *Software Engineering Notes,* 35(3), pp. 8-13 ACM [Online]. Available at: http://dl.acm.org/citation.cfm?id=1764814 (Accessed: 15 November 2012)

Saurin, T. A., Marodin, G. A. and Ribeiro, J. L. D. (2011) 'A framework for assessing the use of lean production practices in manufacturing cells'. *International Journal of Production Research,* 49(11), pp. 3211-3230 Taylor Francis Online [Online]. Available at: http://www.tandfonline.com/doi/abs/10.1080/00207543.2010.482567#.UjTAeNKviSo (Accessed: 29 April 2012)

Schwaber, K. and Sutherland, J. (2011) '*The Scrum Guide - The Definitive Guide to Scrum: The rules of the game.'* Available at: http://www.Scrum.org/portals/0/documents/Scrum%20guides/Scrum_guide.pdf (Accessed: 7 June 2013).

Shewchuka, J. P. (2008) 'Worker allocation in lean U-shaped production lines', *International Journal of Production Research,* 46(13), pp. 3485-3502 Taylor Francis Online [Online]. Available at: http://www.tandfonline.com/doi/abs/10.1080/00207540601115997?journalCode=tprs20#.Uek3ItIwdEI (Accessed: 11 February 2013)

Sim, K. L. and Rogers, J. W. (2009) 'Implementing lean production systems: barriers to change', *Management Research News,* 32(1), pp. 37- 49 Emerald [Online]. Available at: http://www.emeraldinsight.com/journals.htm?articleid=1757547 (Accessed: 2 May 2013)

Sobh, R. and Perry, C. (2006) 'Research design and data analysis in realism research', *European Journal of Marketing,* 40(11), pp. 1194 – 1209 Emerald [Online]. Available at: http://www.emeraldinsight.com/journals.htm?articleid=1579196 (Accessed: 3 January 2013)

Soni, G. and Kodali, R. (2012) 'Evaluating reliability and validity of lean, agile and leagile supply chain constructs in Indian manufacturing industry', *Production Planning and Control: The Management of Operations,* 23(10-11), pp. 864-884 Taylor Francis Online [Online]. Available at: http://www.tandfonline.com/doi/abs/10.1080/09537287.2011.642207#.UjTFy9KviSo (Accessed: 25 June 2013)

Soydan, H. (2002) 'Formulating research problems in practitioner-researcher partnerships', *Social Work Education: The International Journal,* 21(3), pp. 297- 304 Taylor Francis Online [Online]. Available at:

http://www.tandfonline.com/doi/abs/10.1080/02615470220136876#.UjTHItKviSo (Accessed: 7 January 2013)

Stewart, P.; Richardson, M.; Danford, A.; Murphy, K.; Richardson, T.; Wass, V. (2009) *We sell our time no more. Workers' struggles against lean production in the British car industry.* London: Pluto Press.

Stroustrup, B. (2012) 'Software Development for Infrastructure', *Computer*, 45(1), pp. 47-58 IEEE Computer Security [Online]. Available at: http://www.computer.org/csdl/mags/co/2012/01/mco2012010047-abs.html (Accessed: 15 November 2012)

Talby, D. and Dubinsky, Y. (2009) 'Governance of an Agile Software Project', *SDG '09 Proceedings of the 2009 ICSE Workshop on Software Development Governance,* Washington, DC, USA 17-17 May 2009. IEEE Computer Society. Available at: http://dl.acm.org/citation.cfm?id=1569157 (Accessed: 9 May 2013)

Tsoi, H. L., (1999) 'A Framework for Management Software Project Development', *Proceedings of the 1999 ACM Symposium on Applied Computing,* New York, NY, USA 28 February – 2 March 1999. ACM. Available at: http://dl.acm.org/citation.cfm?doid=298151.298486 (Accessed: 14 April 2013)

Vinodh, S. and Aravindraj, S. (2013) 'Evaluation of leagility in supply chains using fuzzy logic approach', *International Journal of Production Research,* 51(4), pp. 1186-1195 Taylor Francis Online [Online]. Available at: http://www.tandfonline.com/doi/abs/10.1080/00207543.2012.693960#.UjTIMtKviSo (Accessed: 25 June 2013)

Wang, X., Conboy, K. and Cawley, O. (2012) 'Leagile" software development: An experience report analysis of the application of lean approaches in agile software development'. *The Journal of Systems and Software*, 85(6), p. 1287– 1299 ACM [Online]. Available at: http://dl.acm.org/citation.cfm?id=2185023 (Accessed: 25 June 2013)

Womack, J. P. and Jones, D. T. (1996) *Lean Thinking: Banish Waste and Create Wealth in Your Corporation.* London: Simon and Schuster UK.

Womack, J. P. (1996) 'The Psychology of Lean Production. *Applied Psychology': An International Review,* 45(2), p. 119–122 Wiley Online Library [Online]. Available at:

http://onlinelibrary.wiley.com/doi/10.1111/j.1464-0597.1996.tb00754.x/abstract (Accessed: 17 April 2013)

Wood, M., Daly, J., Miller, J. and Roper, M. (1999) 'Multi-method research: An empirical investigation of object-oriented technology', *Journal of Systems and Software*, 48(1), p. 13–26. ACM [Online]. Available at: http://dl.acm.org/citation.cfm?id=331352 (Accessed: 10 June 2013)

Yeo, A. W. (2001) 'Global-Software Development Lifecycle: An exploratory study'. *CHI '01 Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, New York, NY 31 March-5 April 2001. ACM. Available at: http://dl.acm.org/citation.cfm?id=365060 (Accessed: 11 June 2013)

APPENDICES

APPENDIX 1 – Interview Questions

1. How does your organization measure the productivity of the employees and their working processes?
    - How does your organization measure the progress when aiming for efficiency?
    - (if they do not have any specific methods to measure productivity) what kind of experiences would help project manager to interpret efficiency?

2. Which project management factors make a difference in the development speed and productivity?
    - What management actions (or “non” actions) would hamper software development speed?
    - What management actions support higher productivity?

3. What is the effect of team size on productivity?
    - Does team size affect the communication and collaboration among its members?
    - Is communication and collaboration of the team members a crucial factor to achieve high productivity?

4. Are the projects with faster development cycles more productive?
    - (If not) what are the decisive factors?

5. How can agile project management methodologies improve processes and methods?
    - Consider a recently completed project and estimate its duration if it had been undertaken five years earlier and with traditional software development methods. How have agile methodologies contributed to the product development speed?

6. Which practices are considered the best to be implemented when developing software?

- Technological aspect
- Human aspect

7. What are some approaches to moving an entire development team to Agile?
    - Which are some challenges faced during this process?
    - Do all members of development teams need to be on agile methods?
    - What is the best practice to move the required people to agile?
    - What are the challenges in doing so?

8. How do managers and their respective teams combine agile and lean production principles in concrete projects?
    - Scrum, Kanban or Scrumban?
    - Challenges?

9. What do managers do for setting and maintaining the right connections among people-processes/principles-technology?
    - What do managers do to make developers committed with their job?

10. Do you have any suggestions on how to improve productivity (in software industry) based on your personal working experience?

APPENDIX 2 – Participants

- Participant 1. **Mattias Gustafsson** - Certified IPMA Project Manager and Project Management Consultant at hiq Göteborg AB, with many years of experience as a Scrum Product Owner and IT Project Manager in the past
- Participant 2. **Stefan Singman** - Professional Scrum Master at Ericsson leading a Scrum team and a Kanban one. Stefan is currently working with agile methodologies in order to find out developing methods and ways of working to improve the quality of the product and the efficiency of all the Scrum teams in the organization.
- Participant 3. **Magnus Jonsson** - Consultant at HiQ Göteborg, uses Agile methodologies to enforce the Test Driven Development and the Continous Interation of features.
- Participant 4. **Andreas Sandberg** - Consultant Manager/Certified Scrum Master at HiQ Göteborg AB, with many years of experience in project management in several well- known companies.
- Participant 5. **Vincent Thavonekham** - Manager at Microsoft Practice & .Net Lab; with past experience as Agile Project / ALM / Cloud Leader , SCRUM coach with Continuous integration, Web project manager, developer, etc.
- Participant 6. **NK Shrivastava** - Management Consultant, Agile Coach and Speaker on PM topics, Springfield, Missouri Area
- Participant 7. **Liang Zhang** - Product Owner and Program Management at Ericsson EAB Stockholm, Sweden, with working experience in Scrum, Kanban, XP, Lean, Test-driven development
- Participant 8. **Paul Oldfield** Team Member at Youmanage HR Ltd, Perth, United Kingdom, with more than 25 years of experience in IT industry, specialized in different fields such as Scrum Product Owner & Certified Scrum Master, RUP, Agile, Process Engineering, etc.
- Participant 9. **Steve Fenton** - Senior Software Developer at Clinical Solutions, Basingstoke, Hampshire, United Kingdom; experienced in agile and lean software development, more specifically XP, Scrum and Kanban to deliver software of high quality.

- Participant 10. **Tim Vermeerschen** - Functional software tester, Gent Area, Belgium, with an economic background from the studies, a broad interest in testing and project management, and a general knowledge about different IT topics
- Participant 11. **Johnny Hermann** - Experienced Lean-Agile Tech Manager for product/service projects & process improvement, Vancouver, British Columbia, Canada; mainly specialized in leading project teams in software and hardware technology development and rollout, especially via Lean-Agile methodologies

APPENDIX 3 – Interview Transcripts

A. Mattias Gustafsson

1. **How does your organization measure the productivity of the employees and their working processes?**
    a. **How does your organization measure the progress when aiming for effectivity and efficiency?**
    b. **(if they do not have any specific methods to measure productivity) what kind of experiences would help project manager to interpret efficiency and effectivity?**

In the company I am working for, they are trying hard to implement the Scrum methodology. They have even changed the business unit, so the line departments are reorganized in cross-functional teams. It is somehow difficult to measure productivity. Especially when you change methodology and you have to start over. In a Scrum world, you need to compare the productivity of the same team for different sprints. Therefore, you have to let certain time periods go, before making any comparison (for a specific team).

2. **Which project management factors make a difference in the development speed and productivity?**
    a. **What management actions (or "non" actions) would hamper software development speed?**
    b. **What management actions support higher productivity?**

I can answer from a Scrum perspective. The role of the project manager in Scrum is undertaken by the Scrum master, who is not team leader anymore. The Scrum master is supposed to support and help the team to clear the impediments so that the team can continue running in the assigned direction. The daily standup meetings, among other things, serve to help the team to increase the communication between each other and to bring up the issues that they are facing during their work.

A really important thing is the follow up work that you do at the end of each important sprint in Scrum, at the end, when you try to identify problems that have occurred or things

that you can be better at for the next iteration. So that time period when looking back and trying to solve problems once and for all, that is the most crucial thing, in Scrum at least.

3. **What is the effect of team size on productivity?**
    a. **Does team size affect the communication and collaboration among its members?**
    b. **Is communication and collaboration of the team members a crucial factor to achieve high productivity?**

I do not think there should be more than 7-8 people in one team. If you need more people you should create another team instead. This leads to a better communication and coordination of the team.

4. **Are the projects with faster development cycles more productive?**
    - **(If not) what are the decisive factors?**

The biggest advantage of having short iterations is that you can change the direction, or check whether the development is going in the right direction. I believe it is very important, because you have to make sure you are doing the right thing in order to succeed.

5. **How can agile project management methodologies improve processes and methods?**
    a. **Consider a recently completed project and estimate its duration if it had been undertaken five years earlier and with traditional software development methods. How have agile methodologies contributed to the product development speed?**

I do not think that Scrum methodology is crucial if you know what you need to develop. When you know what kind of product you are going to create, you can use the waterfall methodology. If it is a more complex project where you do not have all the client specifications when you start, then the Scrum methodology is the key to arrive at the achievement of the goal.
Using iterations is a crucial thing: having points in the timeline that permit you to make corrections according the client's needs. There is another dimension in using agile methodologies. That is to have teams that have been specialized in a certain area. Over time, that team will increase their knowledge and come up with solutions that would not

have been possible otherwise. In that dimension Scrum and other agile methodologies are very good. However, this goes well if you do not change the members of the team.

6. **Which practices are considered the best to be implemented when developing software?**
    a. **Technological aspect**
    b. **Human aspect**

Iterations and self-organization are very important. Iterations' length must be set before the work starts (at least when you have several teams). So you have the same iterations' length for several teams (they start and stop at the same time) in order to make easier the collaboration between them. Otherwise, they will be in different states of the iteration and that could be a hindrance. The iteration length is not decided by the team, but by other people up in the hierarchy, but what is to be included on each iteration, that depends only on the team to decide. Not even the product manager can decide it for the team. The product manager and/or other stakeholders can be present in the meeting and state what they want. For example, a project that is currently being run by the organization I am working for, includes three different levels of product management. It is one person responsible for the whole product; there is a middle segment of product managers that have 2-3 operational product managers, and each operational product manager is in contact with up to three teams, each of them cross-functional and self-organized.

With self-organization comes communication among the team members, and that might be an issue sometimes. It is the role of the Scrum master to fix it. The Scrum master starts with a team and when that team is "up and running", they can start working with Scrum. They know what Scrum means and they can enhance their own work, that Scrum master should take on a second, and maybe a third team as well. In the current project I was talking about, they try to organize "Scrum of Scrum" in order to manage the communication issues. Each Scrum team chooses a representative (i.e. the Scrum master) and all the representatives of these Scrum teams meet 2-3 times per week to discuss about the main issues of their respective teams' work. In other works, in these meetings the Scrum masters bring up the impediments that they cannot solve themselves. In case of such big products and projects, it is only the top level management that has the opportunity to communicate directly with the customer, because having approximately 150 teams makes

it impossible to set direct contact between the developers and the customers. The sizes of the project bring up the necessity of an enterprise Scrum structure.

A third important factor beside iterations and self-organization is the coherence of the team. It is important for the team to stay the same during the whole project.

7. **What are some approaches to moving an entire development team to Agile?**
    - **Which are some challenges faced during this process?**
    - **Do all members of development teams need to be on agile methods?**
    - **What is the best practice to move the required people to agile?**
    - **What are the challenges in doing so?**

I do not think everybody should operate agile. In reality you have certain parts of the development (i.e. those that focus on performance testing) that could be outsourced to a third party (that perhaps cannot participate in different teams). That part is then left out from the teams since it is a specific and separate thing. Therefore, it might be better to set that part up in a traditional way.

The process of moving the required people to agile is certainly associated with several challenges. First of all, you have to change the culture. If the developers have been used to work in their own offices without talking to people, and if you bring those individuals to a shared environment, you have to have interpersonal skills to get that person to communicate.

In order to overcome these challenges, at my organization, managers try to give good examples and to convince people “why this new way of doing it?”. First, organizing large presentation for all the staff and then taking it in smaller groups trying to convince them and to make them involved. This is done by the agile coaches, which have experience in this kind of organizational change.

8. **How do managers and their respective teams combine agile and lean production principles in concrete projects?**
    - **Scrum, Kanban or Scrumban?**
    - **Challenges?**

Both Scrum and Kanban are used in parallel because there are different versions of production running at the same time, often with different customers. One should always handle the defects for the different version of the product. It might be done from the Scrum team, so at the same time they both work on new versions and correct the bugs of the different versions of the product. Some other times there are different types of teams set up for this so pure Scrum teams only work on new versions and there are Kanban teams working to correct faults in earlier versions. Even though these teams usually do not work in the same environment, they all work for the same product. I do not know if there is any mix in one team for the same product, but I think it is more beneficial to have separated teams at this point. In this way, they can evaluate if the methodology is enhancing anything.

Nowadays we use the Scrumfall concept. It is the situation when you have been working on waterfall methodology and you are trying to reach for Scrum but you are not really successful, so you end up somewhere in between.

I think, it would be more appropriate to use one part of a methodology at a time. You begin at the traditional way, then you start using iterations and pair programming. Then it is worth to have some points in time when you reflect on how you have been working so far, but then again it depends very much on how you have been organized before, whether the team have been coherent or not. In our organization, most of the products are organized as projects, but at the end the projects are really the line organization. That means that you get in the same problematic situation when you try to cut up a project and run it in the line organization where you do not have a project team, but just different functions. In that situation, it is hard to get people involved to reflect on what can be improved because they do not work together.

9. **What do managers do for setting and maintaining the right connections among people-processes/principles-technology?**
    a. **What do managers do to make developers committed with their job?**

It is different in different parts of the organizations. One of the most important things, on my opinion, is to have a structure for the requirements. Because traditionally you use to have engineers that are very skilled and know what is needed to be sold. Nevertheless, with the self-organizing teams one needs to know what is needed to be performed, since there is

no more one person doing part of the job. Then you should have some process for the requirements handling. Maybe not to specify requirements for the work in a specific team because the team can do that themselves. However, when you have a common requirement that will affect fifty teams than you need to have a formal process and maybe specific tools where you can write those requirements and where you can get changes to those requirements to the teams as quickly as possible. That means you need to have a more formal requirement handling structure to be able to make it possible for the agile principles to reach the self-organizing teams. In other words, to assign what should be included in a product backlog, from which team members have to pick up tasks on which they have to be working. The most important thing here is for the Scrum masters to have the possibility to change the working processes. As I mentioned before in the ”Scrum of Scrums”, where maybe Scrum masters of ten different teams meet and maybe they conclude that this ordering process is not working as efficiently as we need it to be. Then they must be able to change it. That has traditionally not been the case because their working process has been set by someone higher up in hierarchy and that is impossible for them to change (this has been the case before).

You start with the Scrum meetings that is the base, together with the reflection that you do at the end of each iteration as the main tools to engage the team. At the same time, the Scrum master needs to come up with tools to effect the team members (maybe one to one conversation, maybe a course for certain members of the team, or maybe using experience from some other company to tell the team how they have used Scrum... so all ways to increase the performance of the team are opened). It all depends on the creativity of the Scrum master and on his own team.

10. Do you have any suggestions on how to improve productivity (in software industry) based on your personal working experience?

Trying to think out of the box. It often happens to keep thinking too much about the methodology that is being used, and forget all you need is to have skilled people in the team. Therefore, the best way is to give the right importance to the education and to the increase of the level of knowledge.

B. Stefan Singman

1. **How does your organization measure the productivity of the employees and their working processes?**
    a. **How does your organization measure the progress when aiming for effectivity and efficiency?**
    b. **(if they do not have any specific methods to measure productivity) what kind of experiences would help project manager to interpret efficiency and effectivity?**

One thing that we can measure is the TR – traveler time. That is the internal part of it, and the external one. If a customer has a request about the product and measures the time from the moment that the request is made until it is finished. We fix it if it is still valuable to the customer, or we fix it internally if it is internal fault.
When operating agile, on a team prospective we also measure velocity and notice how it spans over time. However, one cannot measure the velocity in terms of different teams because each team has its own story points. Then it can be about the number of features that are being delivered, but still it is more difficult to measure this way.
We have a three-stage hierarchy of product owners: total product owner, area product owner and operational product owner. The total product owner has 250-300 teams. I am not sure about the amount of teams that a product owner has to manage, but an operational product owner has 2-3 teams and maybe s/he can measure better their performance. I can say how we do it and this is an agile working mode. When a team gets a feature, they have to do a prediction of the whole feature and it can last from six month up to two years. These teams operate agile (even though such a long planning does not sound like agile) because when it is such a huge product, it is very difficult to break it into smaller pieces that give customer value in each sprint.

2. **Which project management factors make a difference in the development speed and productivity?**
    a. **What management actions (or "non" actions) would hamper software development speed?**
    b. **What management actions support higher productivity?**

In my organization we went all in for agile as much as possible, so we removed all the project managers. We have releases of course. We have big ones twice a year. To handle this, the team keeps working on a feature and when they are ready we deliver it to the next upcoming release. We talk to the customer and we plan the feature in advance and we deliver it when it is ready. It is kind of a waterfall method in a higher level of the organization.

An example of what hampers the productivity of the teams: We had a design stop a couple of weeks ago. We have a main track where every team is delivering their codes to this main track. To be able to have this main track with good enough quality with the legacy and everything we have a lot of CI (continuous integration) machinery what needs to test all the time. All the teams add code to the main track and they need many automated tests to secure legacy. A couple of weeks ago we had to design stuff, and that was done from the high management. They decided to focus on quality. It should be the Scrum teams the ones to have the responsibility for the quality of the product. The Scrum masters were not satisfied so they asked top management to not act this way, but to get them involved instead. This kind of actions (the top management one) might hamper the productivity of the teams

3. **What is the effect of team size on productivity?**
    a. **Does team size affect the communication and collaboration among its members?**
    b. **Is communication and collaboration of the team members a crucial factor to achieve high productivity?**

Too small is not good. I am a Scrum master of two teams. One of the teams lost an employee in spring, another one went on parental leave and then we ended up with four people. Since the team members are all parents, with kids and other responsibilities, most of the time any of them might be absent. A team of three-four people is not that good for the discussion in the team. When there are six or seven they have better angles/views to the problems and get better ideas to solve them. In other words, four is absolutely the minimum. Not too many is good either, but I have not found the maximum yet. According to the literature, it has to be between four and eight.

4. **Are the projects with faster development cycles more productive?**
    - **(If not) what are the decisive factors?**

Yes. If I take an overall look at the product, we are releasing more now than before. We are spending less time in integrating and doing final verification. A faster development cycle translates in less time to get to the customer, so it absolutely is more efficient.

5. **How can agile project management methodologies improve processes and methods?**
    a. **Consider a recently completed project and estimate its duration if it had been undertaken five years earlier and with traditional software development methods. How have agile methodologies contributed to the product development speed?**

I believe agile methodologies have improved the working processes because you deliver more, faster and with a higher quality than with the traditional approaches. I see some problems associated with it though. You create these cross-functional teams (because you should not be able to deliver any handovers, you should do everything within the team), so the main issue with this is that you have to learn so much in many areas to be able to do this. It helps to deliver more, but it leads the specialists to lose focus of their specific knowledge, losing their expertise competence. At the same time, it is good to have an overall view and to fully understand the product. Only this way you can be more effective in your specific area. You have to understand the whole in order to get deeper into it.

6. **Which practices are considered the best to be implemented when developing software?**
    a. **Technological aspect**
    b. **Human aspect**

Using Scrum and Kanban is quite beneficiary for the development of a product. The human perspective is also very important. You need team activities to make the team work in unit. It is often forgotten, not in our organization though. My team has just been away for three days in a training about communication and feedback. We went there two days, then we came back for other 2 sprints and then we went again for another full day. Just the team members with the coach conducted many exercises in order to get to know each other and to discuss their issues and their needs, and to set up goals for the team, etc.

7. **What are some approaches to moving an entire development team to Agile?**
    - **Which are some challenges faced during this process?**
    - **Do all members of development teams need to be on agile methods?**

- **What is the best practice to move the required people to agile?**
- **What are the challenges in doing so?**

I think it is necessary to have all the development team operating agile. Once again, we are a very big company and we often depend on others' output in order to start working. If they would not be operating agile, the consequences of their work would weight on us.
Of course, there are challenges associated with the process of moving an organization to agile. I was not part of this organization when they changed, but I know that we went for the Big Bang approach. Change management is an issue because you have to change the whole organizational mindset.
Earlier in time, we had an agile coach helping us. Except from the involvement of a coach, I think it is important to have the support of top management. They should be committed and do a lot of communication with the employees about the organizational changes, in order to get everybody on the same track.
Challenges? Not everybody can be a good team player and those that prefer working individually instead, find it difficult to change their attitudes. They are reluctant to share knowledge and to communicate or work in groups with the other team players. We try to train and coach them, or make them fit into something else.

8. **How do managers and their respective teams combine agile and lean production principles in concrete projects?**
   - **Scrum, Kanban or Scrumban?**
   - **Challenges?**

An organization can have both agile and lean mindsets. They do not contradict each other. Lean is about getting things done as soon as possible, it is about continuous improvements. I am trying to help my team to become more and more familiar with lean. We went to a half-day training to understand the lean principles and I encourage them to read the Toyota book, which I fairly think everybody should read.
There are teams that integrate Scrum and Kanban but I, myself, do not like integrating them within the same team. In Scrum, you have time-boxing for everything (for example if you work in a three-weeks sprint, you have to focus on that for three weeks and you want nothing to disturb you during this time). In Scrum, you are also concentrated and you have your necessary time to focus on that task. Kanban is not like that. Even if you have to focus on your task, everything can change from one day to another. Moreover, the focus shifts to

the new problem. Since our Kanban teams have a supporting role (fixing faults, correcting bugs and errors, etc) their focus can shift from a problem to a more urgent issue immediately. If you are one small team producing a product, you are doing the code, you deliver it, you make a built of it and then you test it. It is difficult when you are so many teams. If 200 teams are working on the same product, then they need automated tests for this. From the time that the development team stops watching the automated tests, it comes to the Kanban team to work on the bugs and other possible errors reported.

9. **What do managers do for setting and maintaining the right connections among people-processes/principles-technology?**
    a. **What do managers do to make developers committed with their job?**

The managers try to communicate rightly the component's description to the team through the help of the guardians. The guardians are people integrated to Scrum/Kanban teams but they help to introduce and train the team to work on the specific components.

10. **Do you have any suggestions on how to improve productivity (in software industry) based on your personal working experience?**

I think I said them already. The biggest need is to focus on the human aspect. It is easy to forget about that when working with technical issues. The teams need to be well organized and managed in order to deliver.

To implement Scrum and lean you have to refer to the books, but strictly following those versions makes work difficult sometimes. Therefore breaking the rules can help in special occasions.

C. Magnus Jonsson

1. **How does your organization measure the productivity of the employees and their working processes?**
    a. **How does your organization measure the progress when aiming for effectivity and efficiency?**
    b. **(if they do not have any specific methods to measure productivity) what kind of experiences would help project manager to interpret efficiency and effectivity?**

We measure the velocity to make sure that the teams do what they do and that the estimates are correct and then based on that, the teams themselves take action. (”take action” in this case means that the team try to improve. After each sprint, they have a meeting and discuss how can they be more effective. The teams themselves want to be more effective, that is a reason why they work a lot with themselves. Management doesn’t have to interfere, except in some cases. Otherwise it is good to let them be as effective as they can.)

In addition to velocity, there are soft values as well. For instance, the delivery of the right features. We have to have the right customer contact during the sprint in order to give them business value after the sprint. Each feature that the team delivers should be what the customer wants, so it is very important. When we deliver a feature after a sprint, we let the customer use it, so if they are not satisfied the team makes further changes to it to align it with the needs of the customer. All this process happens before the final release.

In a summary of the question, it is all about velocity and the customer satisfaction.

2. **Which project management factors make a difference in the development speed and productivity?**
    a. **What management actions (or “non” actions) would hamper software development speed?**
    b. **What management actions support higher productivity?**

The most important is that management lets the team be self-organizing and that they provide the teams with what they need (i.e. a prioritized backlog). Self-organizing because if the teams are allowed to make decisions themselves, they are more motivated.

Management should only discuss what should be done (having customer talks initially and then discussing what the team should be done), then the team should go on discussing how the things are implemented, how the architecture should be done and so on. The teams should be also allowed to have customer contact directly, not by a proxy management and other similar ways. Sometimes the product owner draws up the big sketch, the initial ideas on what the customer wants, and after that, it is the team who interacts with the customer. On my opinion, it is more effective than having a part in between the customer and the person who is working on the feature, because otherwise it becomes like the whispering game: things get distorted along the way.

3. **What is the effect of team size on productivity?**
    a. **Does team size affect the communication and collaboration among its members?**
    b. **Is communication and collaboration of the team members a crucial factor to achieve high productivity?**

I would say eight (plus/minus two); this is a good team size. Because if it is a big team, you get too much overhead within the team, and you get teams within the team, but somewhere between 6-9 people are an optimal size. This is good for the communication among team members as well. In case of too large teams it is difficult to find a room where to sit in, and one can miss out what is happening in one side of the room. It is good if all the team members are located in the same place. This way they can talk to each other without disturbing other teams, and most of the times, discussions that happen within the team are work related, and it might be important that the other team members hear the discussions even though they might not be participating in it.

4. **Are the projects with faster development cycles more productive?**
    a. **(If not) what are the decisive factors?**

The sprints within the same project should be equally long. It would be more efficient to avoid having i.e. a two-week sprint, a four-week sprint, etc., but instead agree on a sprint length in the beginning of a project and stick to that decision for the rest of it. Two-week sprints happen to be too short, hence three to four-week sprints are optimal, and five-week ones are too long. The development cycle is somehow related to the team size. If you have a team compounded by two-four people, two-week sprints would work well, but more than

six people per team overhead costs for each sprint, as there would be too many people working on the same tasks. Coming back to the question, for sure shorter development cycles are better than the longer ones; just the sprint concept is definitely better. This is because of the fact that the team has a chance to deliver and get the customer feedback in a shorter time. For example, if you work on a sprint for three weeks, by the end of that time you get to know that you are on the right path, but if you have to work six months and get to know by the end that it wasn't the right track, the whole time would consist in waste.

5. **How can agile project management methodologies improve processes and methods?**
    a. **Consider a recently completed project and estimate its duration if it had been undertaken five years earlier and with traditional software development methods. How have agile methodologies contributed to the product development speed?**

Agile methodologies have improved a lot the working processes. I have a very good example of a project on which we were working some times ago. We used to work with waterfall, and we were basically too late to the market, we did not hand in to the customer what they wanted, and when we gave them what they wanted, they had already changed their minds, because internet came up in this era and everything should be web-based instead of just a piece of software. So first, we were too late to the market, and second, we didn't get enough customer feedback during the process. Furthermore, we had huge integration problems. Then we started to work with Scrum in a very small project that was specific to one country; we had the quick iterations, sprints, we got the customer feedback all the time. We saw the advantages and we noticed that we could implement that in bigger projects. So we started to implement it in a very large project in India, Sweden, Norway, Germany and U.S., where the implementation teams were located. There were in total twenty-five Scrum teams at the edge. It was very, very effective as we increased not only productivity, but we delivered also the right things and the quality went up dramatically. So by the end we delivered a much, much better software. To sum it up, agile development methods showed to be able to improve the delivery time, the interaction with the customer and the quality of the work.

6. **Which practices are considered the best to be implemented when developing software?**
    a. **Technological aspect**

b. Human aspect

Scrum in a combination with test-driven development, in order to get a lot of automated tests and continuous integration, so it helps to integrate during all the time one team's work with another team's work. As for the human aspect of the question, firstly, I think management should ensure the promotion of sprint successes by organizing sprint celebration, and the promotion of knowledge activities so that the team members feel like they are allowed to absorb new knowledge. Both these practices are important for the motivation of the team. In practice, usually this is done in a round-back session. The team members have a lunch together (this does not have a big monetary cost for the company) and a small lecture about a new topic or technology. Secondly, the team should be self-organizing. The team should be allowed to take decisions about their working processes instead of having a manager coming to them and telling them how the tasks should be done. Thirdly, management should promote and support the discussion among the team members, for coordination purposes.

7. What are some approaches to moving an entire development team to Agile?

a. Which are some challenges faced during this process?

b. Do all members of development teams need to be on agile methods?

c. What is the best practice to move the required people to agile?

d. What are the challenges in doing so?

One of the biggest challenges, is getting people who are used to have their own offices, to sit in a team room and share the same space. Then of course learning the Scrum principles and getting used to the Scrum methodology is a big challenge because Scrum is a different mindset. This challenge might be overcomed with the help of the agile coaches who will teach them and make sure that everyone knows how to work. Then one should look into management and the challenge that they face when jumping from ”leading work all the time” to ”supporting the teams”. Usually management have a hard time to going to agile because managers have to resign from their previous respective roles. We face such a challenge with the US managers who were quite reluctant to move to Scrum. However, by the end they adjusted and adapted and we could see the positive results.
Not necessarily all the members of a team need to be moved to agile. However, everyone involved within projects should be knowledgeable of agile; they should have some kind of role within an agile team. Around software development there are a lot of supporting

activities that need to happen. For instance, IT department people don't have to be agile; they can work as they used to work before. Of course, it is good if they can move to Kanban, since the job of IT department is mostly maintenance and if they adopt to Kanban they can be more action driven.

When moving to agile it is advisable to train the people involved in this process. We did a mistake years ago as we didn't undertake any training, so we basically trained ourselves and it took us some time to get used to work with Scrum. If we would have agile coaches, as we nowadays do, we would experience a faster and a more effective process of moving from waterfall to Scrum.

8. **How do managers and their respective teams combine agile and lean production principles in concrete projects?**
    a. **Scrum, Kanban or Scrumban?**
    b. **Challenges?**

On my opinion, Kanban would be suitable for the “after market”; once you have delivered the software, you it helps to look after defects, small enhancements and so on. If you are familiar with Scrum and you have some knowledge about how Kanban works, it would be easy to integrate Kanban processes, especially to handle defects. If you have a team that is delivering sprints in a project, and after they release them they get some small defects on it, and if they used to spend some time on fixing them, now they can do it with Kanban. The ideal would be to have several teams within a project, lets say a total of six teams for example, five of which will focus on developing the software; then you take one team member from each team and put them in a sixth team. In this way it will be a team of five persons that will work only on defects. This means that the other teams can continue to work on new development and new development only, and at the same time there you have a team working in parallel with them to fix bugs.

9. **What do managers do for setting and maintaining the right connections among people-processes/principles-technology?**
    a. **What do managers do to make developers committed with their job?**

I would like to answer the question from the Scrum point of view. Working with Scrum is often entertaining for the team because they get to make decisions themselves, or for example people that would like to work with databases but they would like to, get a chance

to do it within the team perhaps. This motivates the team members to be more committed and to reach the goals for the sprint. Otherwise they wouldn't like to work with Scrum. So it becomes a responsibility for the team members to reach the goal. Then after each successful sprint should be celebrations. In our case, it can be anything such as go out for a beer, buy a team cake, do some funny activities. It doesn't have to cost much, but just doing something out of the ordinary instead of being in the office. I think it is very important because working with software development projects in general is tough and sometimes it brings out feelings in the members that it never ends. Scrum has changed this mindset with its "sprint" concept, that they have to stop time after time and to experience some satisfaction of reaching the goal. If management provides them some small little incentives such as a beer, an ice-cream, a meal they will feel more pleased and motivated for the next sprint.

10. **Do you have any suggestions on how to improve productivity (in software industry) based on your personal working experience?**

The basic rule should be that everyone should work on what they are best on, so for instance management is good at making decisions and handle the business and the teams are good at developing software. So mainly, this should be the way it works: teams focused on developing software and management focused on supporting the team and adding quality to the business. Other than that, I think it is mostly about motivation and elimination of the useless work activities, i.e. unnecessary meetings. Let people themselves judge if they need to attain certain meetings or not according to the extent to which they find them effective. In this way management meetings have to be effective; they should not be too long to avoid the feeling of a waste of time and to set the standard to the employees to keep their own meetings short and effective. The last but not the least to motivate the team members to commit on what they are doing and make it possible for them to have customer contact.

D. Andreas Sandberg

1. **How does your organization measure the productivity of the employees and their working processes?**
    a. **How does your organization measure the progress when aiming for effectivity and efficiency?**
    b. **(if they do not have any specific methods to measure productivity) what kind of experiences would help project manager to interpret efficiency and effectivity?**

We are struggling to find the right KPIs (key performance indicators) to measure the productivity of the teams. We have counted the number of the improvement features that the team is delivering while working with development tools. For other teams that are working on product development, we have not had feedback on regular basis as the features that they were developing were very long. As a result, we it is difficult to get the data often. This is why we are struggling to measure the productivity of the teams. Secondly, since we are working with sprints, we check if the user stories are delivered successfully. Indirectly this is some kind of productivity measurement.

2. **Which project management factors make a difference in the development speed and productivity?**
    a. **What management actions (or "non" actions) would hamper software development speed?**
    b. **What management actions support higher productivity?**

We are working fully according to Scrum so we try to embrace its features as much as possible. As for the human factors, the people skills, those are already built-in in Scrum. We have a clear purpose what to do, and that gives a good motivation, and an aim to improve all the time (as part of the agile philosophy). That means that if an individual improves the own technical skills, he/she will also feel good at work. The improvement is measured mostly based on a soft evaluation that in a hard one. If there is a feeling that something needs to be improved, the team learn from the workgroup or set up learning sessions about certain areas. The teams are totally self-organizing and are given the freedom to choose who they want to work with, according to the common benefit of the group.

3. **What is the effect of team size on productivity?**
    a. **Does team size affect the communication and collaboration among its members?**
    b. **Is communication and collaboration of the team members a crucial factor to achieve high productivity?**

We had a team with 10 persons in the beginning, but we got the feeling that the teams compounded by 4-5 individuals leads to less coordination and communication efforts and as a result to a higher productivity.

4. **Are the projects with faster development cycles more productive?**
    - **(If not) what are the decisive factors?**

We have not experimented that much the cycle lengths. We have been running two- or three- weeks sprints and this length has shown to be good for us. Generally, we use the same length of sprints within a project.

5. **How can agile project management methodologies improve processes and methods?**
    a. **Consider a recently completed project and estimate its duration if it had been undertaken five years earlier and with traditional software development methods. How have agile methodologies contributed to the product development speed?**

I have been working with both the traditional (waterfall) and agile methodologies and my experience is that the agile approach improves a lot the working processes in various aspects such as motivation, coordination, communication, risk handling, the value of what is produced, etc. The only drawback is that when individuals of the big organizations are brought to work together in Scrum teams, some of them find themselves too challenged, mostly emotionally, as for example when they lack the necessary skills to keep up with the pace of the others. In the traditional methods this is not visible. Therefore, such individuals find those methodologies more appropriate.

6. **Which practices are considered the best to be implemented when developing software?**
    a. **Technological aspect**
    b. **Human aspect**

I have been working with Scrum, and partly with Kanban. Agile methodologies have proved to be the best. We are using Kanban with one of the teams because the backlog is more difficult to organize and keep for more than a couple of days. Sometimes this team does service work too. This is a trial for the next coming months to see if this method shows to be successful.

In a developing team everyone needs to have an aligned process. That could be Scrum, Kanban, maybe sometimes Scrumban.

7. **What are some approaches to moving an entire development team to Agile?**
    - **Which are some challenges faced during this process?**
    - **Do all members of development teams need to be on agile methods?**
    - **What is the best practice to move the required people to agile?**
    - **What are the challenges in doing so?**

All the members of a developing team should move to agile. We did the switching process like a big bang, though it was a small department. The engineers themselves felt that they needed another way of working, so they mostly inspired it, and then the management caught on and tried to cope with it. It was mostly about reading (home studying) and daring to try. We started with one-two sprints in one team and then all of the other guys switched at almost the same time. It was a department compounded by 35-40 people. In other words, only by self-studying and practicing we introduced Scrum to that department. We did not face any big challenges as before that the organization had experienced a huge IT project failure, so we all were motivated and willing to try any new way of working that might show to be effective. As long as you do not have people that are against the change, the challenges as not very serious. Now it is more advised to switch to agile methodologies step by step, from piloting first and then go off, trying to implement Scrum in a team and if that shows to be successful, to move to the next team. However, I would advise the developing team to run according to Scrum because my experience has shown that it is a superior way of working comparing to the traditional one.

8. **How do managers and their respective teams combine agile and lean production principles in concrete projects?**
    - **Scrum, Kanban or Scrumban?**
    - **Challenges?**

We should be able to develop the processes we need. I have seen a team using working progress numbers to measure the ongoing tasks of the team members in a project that was running based on a Scrum framework. This is about setting limiters: for example “how many ongoing tasks you have in a team” so that the team elaborates on that. In this way, each time they are sure that they are not overloaded with work and makes it clear for the stakeholders around the team, what the team members are working with. That is not incorporated in Scrum, it is more Kanban, but if you can combine those you use sprint lengths, the visibility with the Scrum board, and you can have new releases with a certain number of corrected troubles reported, prioritized tasks and a number of deliveries expected to be achieved at a certain point in time. That can be a successful implementation of Scrumban.

9. **What do managers do for setting and maintaining the right connections among people-processes/principles-technology?**
    a. **What do managers do to make developers committed with their job?**

We have tried to keep the team inspired by making clear the group goals and the product goals. We have done exercises to share the experience and the points of views of the team members. The idea is to create a motivating working environment. The team members should be clear about what are they expected to deliver and when do they have to deliver it, and get the feedback. Feedback is important as it is constructive. The individuals that feel challenged (keeping in mind the motivation part) have problems to provide feedback, particularly in the teamwork. I aim to make the goals and challenges clear to my team, to provide them with feedback from other stakeholders and to inspire them to provide feedback for their colleagues, the other team members.

10. **Do you have any suggestions on how to improve productivity (in software industry) based on your personal working experience?**

Two practices are retained as the best for software development:

- Motivation driven by clear goals
- Working with competence

E. Vincent Thavonekham

1. **How does your organization measure the productivity of the employees and their working processes?**
    a. **How does your organization measure the progress when aiming for effectivity and efficiency?**
    b. **(if they do not have any specific methods to measure productivity) what kind of experiences would help project manager to interpret efficiency and effectivity?**

In our case, we measure that through graphical management, visual management, so we create burn-down charts during the sprint and then, at the end of the sprint, we measure the velocity of the team. In this way, from one sprint to another we follow the progress of the velocity. Another measure, which consists on measuring the time of the bug fixes, is a very important one as it shows the maturity of the workflow. The third one is the feedback from the customers, which determines the velocity of the team. For example, if a team has been given ten user stories to complete/finish, and only nine of them are validated by the product owner you will have the first user story saying that you have ten points, the second one twenty and you sum it up in the end. The result gives the velocity. Otherwise, it is quite difficult for us to measure the satisfaction of the customer, if we do not have such a measurement. It is difficult to get this measurement from the product owner because whether he/she is too busy to do it, or s/he feels that it is not important for her/him to provide such feedback.

2. **Which project management factors make a difference in the development speed and productivity?**
    a. **What management actions (or "non" actions) would hamper software development speed?**
    b. **What management actions support higher productivity?**

I would like to mention two kinds of management styles: the macro-management style and the micro one. Some of the people in my team prefer the micro-management style because they arrive in the morning and they find a very detailed list of the tasks that have to be accomplished during the day, so they just have to sit down and follow the rules. The rest of the team members, those who like their freedom, prefer the macro-management style.

Agile management approaches are more appropriate for these people as it gives them the vision for a given sprint, the vision for a given product and the one for a given date. This is the reason why every day we have stand-up meetings so that each developer is free to act within that day according to the vision that he/she has to complete. Therefore, in an agile world he/she is free to organize his day and do all his tasks. Agile methodologies are more rapid, because whenever the person has got an impediment, he/she will try to find a solution by him-/herself, think on its own and discuss it with other persons.

While the first group of people that like the micro-management will just sit there and expect the management to take a decision and remove the impediment. These people are not active in such situations. To ensure that this is true, we organize agile games or innovation games that prove through experiments that the style of management that provides people freedom to think and act is more rapid to solve e given problem. Some of the people will not be suitable for agile methods, either in terms of developers or in terms of managers. If they do not have the agile state of mind, it will not work.

I would like to mention the case of a person who loved to be micro-managed. She was not working at the same pace as the other people who enjoy their freedom. She was not doing much stressful work as the other people used to do in their autonomy, trying to be critical when fulfilling their goals during the day for the sprints or for the entire project. What we could do to fix that, was to have stand-up daily meetings, so people could share their stress and perceive the extent to which they had to be responsible (how much their actions would affect those of their colleagues), considering the fact that they are part of the whole team.

3. **What is the effect of team size on productivity?**
    a. **Does team size affect the communication and collaboration among its members?**
    b. **Is communication and collaboration of the team members a crucial factor to achieve high productivity?**

When I got my Scrum master certification, we were taught that the ideal team size is seven people plus/minus two. That means that those teams compounded by twenty people may slow down the work of the team because autonomy and self-organization is more difficult due to the huge number of the team members. Communication will also be difficult. On the other hand, if you are part of a too small team, there is no point in embedding Scrum and following its artifacts, as it will but slow down the work processes. For example, if you are

two-three people in a team, there is no point to have stand-up meetings every morning. However, it is possible to be agile without being Scrum.

4. **Are the projects with faster development cycles more productive?**
    - **(If not) what are the decisive factors?**

I came across a project where the development team was releasing code every day (with one-day sprints), which is a very short development cycle, so they had to organize themselves and be really mature when deciding who is doing what, who is testing what and also they should have a software factory. So whenever a developer committed his/her codes, an entire machine had to be there to check the codes, to ensure the quality is the right one and to send an email to the person who would be responsible for testing that piece of code. Therefore, one-day cycles are quite productive, but you have to be very mature to reach that cycle-length. However, this type of release is only suitable to only some editors; not every team can do that. In this case, we are only seeing one version of the application, which was once released and then used by everyone afterwards.
Whenever you have a large sophisticated scenario, you cannot rely on one-day cycles, therefore you should consider a two- to three-week one, as two/three-week iterations are the best. This is because some of the checks that you have to do go through a long validation process which cannot fit within one day.
I would like to come up with the example of a project running in a medical environment. A two- to three-weeks development cycle is long enough to provide the necessary time to follow the rules and regulation, to sign the contracts, etc. whenever a new customer is on board. Another factor that should be considered is the existence of the other teams throughout the world. For example, we had this project being run in France, India and USA because of the size of the project. In India, they used to follow the three-week cycles, whereas in France we enjoyed the two-week cycles. At some point we had to align ourselves, so we chose the three-week cycles in order that everyone could follow the same pace and synchronize. For the synchronization, we had to do what is called “Scrum of Scrum”. You apply Scrum when working with a team with seven (+/- two people), and then you take one person from this Scrum team (let us say from France), another person from another Scrum team (let us say from India) and the two of them do another Scrum meeting. In this way, if something goes wrong, it is easier to locate and correct it.

5. **How can agile project management methodologies improve processes and methods?**

a. **Consider a recently completed project and estimate its duration if it had been undertaken five years earlier and with traditional software development methods. How have agile methodologies contributed to the product development speed?**

We should consider very large projects, not only the development part of them, but the whole phases from the gathering of the specifications up to the end, when we retire the application, in order to see the role that agile methodologies have played to their improvement. To retire an application means to remove/kill it. When customers retire their applications and want all their data to be deleted, or withdrawn in order to use it internally, the process should be thought from a general perspective, the whole project should be considered.

For long-time projects, i.e. 10 years, it is very difficult to sit down and think about all the steps and workflow for the next six-months or one year. Agility offers us the ability to start very rapidly and in parallel. For example, (the scenario that I was describing) let us start from gathering of the needs, development team, delivery, production, etc. It is a long cycle. You can start all of them from day one. Even without any specification. I started a new project yesterday, without any specifications. We started to build up the software factory, like an assembly line when you are dealing with cars. We put all the robots together, without even knowing what kind of car you will be building. You know at some point the basic things, but not the specifications. The customer updates time after the time the programmers with his requirements and every two weeks we can see the progress. This is beneficial as if we observe that something is not going right, then all we waste is two weeks of work instead of years of time (according to the traditional methods).

6. **Which practices are considered the best to be implemented when developing software?**
    a. **Technological aspect**
    b. **Human aspect**

As we are talking about agility, in terms of the methodology, the practices may be guiding practices and software factory ones. Software factory is the key to the success of the application. I will quote another agile method, XP, within which you get the concept of software factory. Software factory is a kind of robot (the assembly line) that checks every single time the quality of your code. The best software factory is the one that we call

“unbreakable build”, which means that the developer codes and tries to submit this code to the repository. If the quality is not high enough the code will be rejected by the robot. This is also called “gated check-in”. Therefore, you have the methodology and the tools. This relies a lot on tooling. In our case, we use only two tools since I am working on Microsoft technologies. We use Visual Studio and Team Foundation Server 2012. Whereas in a Java world, you have to mix many different tools. You have tools that do the repository, others that do the continuous build; others do the continuous delivery and so forth.

Concerning the human aspect, agility and more concretely Scrum and the roles that it involves (the development team, the Scrum master and the product owner) ensures that everybody works with the highest velocity and that there is coherence and harmony among the team members.

7. **What are some approaches to moving an entire development team to Agile?**
    - **Which are some challenges faced during this process?**
    - **Do all members of development teams need to be on agile methods?**
    - **What is the best practice to move the required people to agile?**
    - **What are the challenges in doing so?**

I have done this in the past with various teams so I can quote various examples. I will quote the happiest conversion principle: the fact that we created the product development team from scratch. I knew about agility but nothing about Scrum. Everyone was willing to learn. I was there to recruit the development team members, so I had to detect if the team members were enjoying the micro-management style or not. If that were the case, I would not ask these people to join our team, even if they would be very experienced and capable on using technology. Therefore, the team was there very willing to learn about agility (same for the Scrum master and the product owner). I was there as a Scrum coach. This was from the bottom-up.

As for the top-down approach, the top management was helping us to succeed in implementing this methodology. They ensured all the development team to be located at the same place, instead of a developer sitting somewhere in a free chair, and another one in another free chair in another floor. The collocation at the same place and the visual managements on the wall made the work more productive. Agility made us able to prioritize our tasks and find shortcuts to solve the problematic situations, throwing 1/3 of the unneeded specifications.

Another scenario was a multinational project in the healthcare industry. Implementing agility there was more difficult as we could not bring at the same place the whole teams operating in different countries.

Do all the developers need to be operating agile? If the members of the agile teams have to cooperate with others who do not operate agile, this might lead to conflicts, so it would be ideal if everybody operates under the same conditions. It often depends on the "micro-management" addiction of certain people, who need to be more into the macro-management practices to ensure the success of the work.
As for the challenges, for me personally, other team members' attitudes such as spending time on social networks such as Facebook, Twitter, LinkedIn, etc. was a challenge in the beginning, until I came to discover that if I prohibit people to open those websites during the working hours would decrease their productivity instead of leading to better results. I discovered that if I let them free to organize their time the way they want, they would be happier and more efficient. I realized that some people need coffee, some people need to spend time on Facebook, etc., but as long as they accomplish their tasks and make the customers satisfied by delivering the expected outcomes, the way they choose to organize their time is not an issue for me.
Another methodology is the copyright action type. This methodology comes from the top sportives in the world. The idea is that as top sportives need to eat healthy, train and prepare for competitions while having a coach following their activity and adjusting accordingly to their state, in the IT world we do not do it at all. It is a bizarre. If people call you at 3:00AM in the morning asking why an application does not work, we are losing millions per hour, etc., it is really as stressful as a sportive match. We have to consider that the IT guy, as a sportive, has to do his drugs - let's say for example "Facebook" for five minutes, etc. Another person might not be there due to family problems, etc. The idea is to consider the human aspect of the team member, On top of that, I brought the idea of the sportive coach because they are able to perceive how a person feels by closely looking into the way they feel and act. In the IT world, if you get the vision of a team member, you can lead him/her to better working results.

8. **How do managers and their respective teams combine agile and lean production principles in concrete projects?**
    - **Scrum, Kanban or Scrumban?**

- **Challenges?**

Scrum is like the extra wheels of a bicycle that help you to learn how to ride. This way Scrum helps to understand the agile philosophy through using different techniques such as time-boxing, burn-down charts, etc. Kanban is also an agile methodology but it has less constrains than Scrum. We have no practical experience in using lean methodologies, but I have met a lot of people who are using Scrumban, a combination of Scrum and Kanban approaches. Therefore, even though I haven't had the chance to implement Lean in my working practices so far, I am a big fan of the lean philosophy.

- think big - think upon until retirement of an application from the very beginning of a project
- act small - deliver in small pieces periodically
- fail fast - find major problems rapidly
- learn rapidly - adjust your vision

9. **What do managers do for setting and maintaining the right connections among people-processes/principles-technology?**
    a. **What do managers do to make developers committed with their job?**

Heavy tooling and management are the two important things, but they should not be the aim. The tool should be suitable. The tool is an impediment, so that the reason why you get the agile games and the innovation games, i.e. the speed boat (you draw a boat in the wall, where it is and where it wants to go). Agile games and innovation games serve to point out the impediments that the tools may create.
Management is also important so it has to be open-minded and get rid/through away what does not serve to the development team. The solution is to keep the standards high and find the way to them.

10. **Do you have any suggestions on how to improve productivity (in software industry) based on your personal working experience?**

Do small things to make the team happy i.e. sprint celebrations. At the end of the sprint, if we perform well, you can do everything but work for some minutes. For example, buy pizzas, spend time watching videos from TED conferences, etc. We do this to consider the pace and the rhythm of the team. You cannot run forever what is written in a programming

file. It runs forever and has no break. However, as stated above, like a sportive, an IT developer should be prepared, started and celebrated when it achieves some results. Other suggestions is to do games that develop the creativity of the team, i.e. the spaghetti and marshmallow challenge. Last but not least, is to consider change management, ask for feedback and try to be opened to it.

F. NK Shrivastava

1. **How does your organization measure the productivity of the employees and their working processes?**
    a. **How does your organization measure the progress when aiming for effectivity and efficiency?**
    b. **(if they do not have any specific methods to measure productivity) what kind of experiences would help project manager to interpret efficiency and effectivity?**

    I normally use a scorecard. I put the scorecard in a dashboard and I measure several categories of metrics:

    - Variances: Planned vs. Actual (In terms of efforts, duration of the tasks etc.). Earned Value Analysis is used to measure the performance of the teams.
    - Productivity trends – the amount of software produced in one-person-day or one-person-hour; the number of function points per hour/day; the number of story points per hour/day. These parameters are calculated across various projects and then an analysis is made in order to observe the productivity of the team in a month or three-month trends.
    - Quality is based on three parameters:
        - The level of rework – fixing bugs and other problems – calculated by repaired-to-build ratio which should be less than 5% and which is calculated by the number of hours spent in fixing defects dividing the total number of hours spent in accomplishing the task (producing something). If this ratio is more than 15%, then it is alarming. It is important to point out that the number of defects is not an objective measure, as when you compare two teams that might have gone through the same number of defects, it is difficult to compare the level of the respective problem sizes.
        - Changes (Change-to-big) – depends mostly on the quality of the requirements produced: the clearer the requirements, the fewer the changes.

Those are the measures that I use. I put them in a scorecard, which goes in the dashboard and becomes available to everybody. We measure each month to see the trends and judge if they are favorable as the time passes.

To measure productivity you need to have a baseline measurement like either you say (in case of software) function points per person month, or story points (the velocity – the number of story points produced by the team). In case of using story points, the team should be the same in order to be able to make comparisons and to have a true picture of the team productivity. The story point is a subjective way to measure productivity, there is no guide. It is different with function points. You adopt processes and methodology to calculate function points and those processes can be adopted across different teams, different locations, etc. so that is independent. Basically, you decide upon which parameter to use for measuring productivity and then you proceed with measuring it.

2. **Which project management factors make a difference in the development speed and productivity?**
    a. **What management actions (or "non" actions) would hamper software development speed?**
    b. **What management actions support higher productivity?**

If the management ignores the processes that the team should follow (i.e. if the management does not give importance to signing in the requirements before sending them for design and development); if the management does not listen to the team; if the management replaces the tasks of the developers after a project has started, changing the priorities, etc. it leads to dissatisfaction and confusion in the team. That hampers the development speed.

Being respectful, understanding the processes and avoiding unnecessary changes, managing the portfolio, etc. leads to higher productivity. A very important point is to empower the teams to take actions, make decisions, do mistakes so that the team will learn and get motivated. Motivation is the key to success.

3. **What is the effect of team size on productivity?**
    a. **Does team size affect the communication and collaboration among its members?**
    b. **Is communication and collaboration of the team members a crucial factor to achieve high productivity?**

Too big teams affect the productivity negatively. There is a formula to calculate the communication channels. It is based on the number of stakeholders (N): **N*(N-1)/2**. For example if a team has 10 members, there will be 45 communication channels. If you have a double sized team (20 members) there will be 190 (5 times more) communication channels. More efforts are required to communicate and keep everybody at the same pace. Because of the inefficient communication, there can be a lot of things that could go wrong in a project. A team of more than 8 people requires extra efforts to coordinate. Too large teams reduce productivity.

4. **Are the projects with faster development cycles more productive?**
    - **(If not) what are the decisive factors?**

The projects with fast development cycles not necessarily are more productive. It depends on "how fast". I think the development cycles should be at least two weeks. The team needs some time to get the things in place in order to say whether the cycle is productive or not. I have in mind two projects. One of them had the two-week development cycle and the other had the four-week development cycle.
If you have a dedicated team at one place, working on a single project, then the two-week development cycle works fine. In contrast, if you have a team working on several multiple projects, not collocated at one place (they are spread over), then you need to provide some level of advanced planning, ex. "what people are going to do", and if that changes too fast, then it becomes a problem. Therefore, two-week cycles are not appropriate in this case. I have seen the team struggle when they have to be multitasking and to work on multiple projects.

5. **How can agile project management methodologies improve processes and methods?**
    a. **Consider a recently completed project and estimate its duration if it had been undertaken five years earlier and with traditional software development methods. How have agile methodologies contributed to the product development speed?**

Agile and traditional methods are two different categories that are both strong in their own environments. If we consider the example of an application migration project, or a just a data migration project (somebody wants to move from an old legacy database to oracle), it can be very well realized using the traditional methods since the requirements are fixed and

well defined. If we consider the example of building the website of a company, the requirements may change as you go. It is very hard for everybody to spell out each and everything about a website. One might want a website of 50 pages and he/she might want it to be interactive (not to contain only narrative content). Still it is not clear what content should be placed on each page. Still if there is the basic structure and an overall idea about the services that it should offer, it is enough to start the work in the beginning and change the content afterwards. As you go, you can keep changing and involve the customers. So agile works well in those cases when one does not have clarity, can deploy systems in an incremental fashion, and involve customers. There is a blog: “Is agile a silver bullet?” in which people discuss about situations where agile works well. In other words, agile and traditional are two different methodologies, which can both succeed in different project environments. Which of them is more successful cannot be said for sure; it all depends on the nature of the project and on the environment in which the project is being run.
I cannot have a right answer if agile methodologies have improved the software development processes and methods. Waterfall and other traditional methodologies have their own processes. If those processes are executed well, they can result on successful projects. The same way agile methodologies have their own processes. If those are not effectively done, they can be disastrous. So it is not a question whether agile has improved the processes or not. They both have different sets of practices. They are two different methodologies. Whatever methodology you pick, if not applied properly, it might result in a failure.

6. **Which practices are considered the best to be implemented when developing software?**
    a. **Technological aspect**
    b. **Human aspect**

- One of the best practices to be implemented is having clarity about the requirements before starting the work/coding (no matter the methodology). In agile one might have fewer requirements at the very beginning.
- Test faster. Test as fast as you can and let the development and the testing team work with the requirements. I am working in a project. I am an agile coach and I am coaching the team to adopt agile practices and I am doing the retrospectives. For the main sprint that we did, we also did many tests that reported a lot of issues, a lot of defects. The

result was that some of the other work that they were supposed to do, could not be completed and got pushed to the next sprint. Therefore, my question to team was “why did you have so many defects?” and they responded “because the requirements were not clear”. Now they have a requirements’ document and we organize joined requirements review sessions to get to know and sort out the differences. The development team, the managers and the business analyst need to have a common understanding of the business requirements and this is the third important point.

- Have a joined review so everybody has a common understanding.

7. **What are some approaches to moving an entire development team to Agile?**
    - **Which are some challenges faced during this process?**
    - **Do all members of development teams need to be on agile methods?**
    - **What is the best practice to move the required people to agile?**
    - **What are the challenges in doing so?**
        - Do it in steps. There is another blog called “Implement the agile PMO in an agile manner”
        - Planning
        - Sprint review
        - Daily stand-ups
        - Do the retrospectives and learn from them.
        - Have a backlog created

In the project that I am working with, I also needed a product owner. The team suggested the product owner to be the president of the company. I denied because the product owner should have time to spend with the team, should be close to them. Then we found a good business analyst, who could play this role. It is good to make sure that all the roles are defined and that there are people responsible for the different roles. Then involve the team to do more and more by going step by step. Do something new every month.
One of the fundamental challenges is the mindset. The company where I am doing the agile coach will go and commit a date with the costumer. Such a thing you can do only when you have absolutely clear requirements. In agile you do a release planning at a high level but you do not do much detailed planning in the beginning, because the idea is that the things will be changing. The development team says: “If we do not plan, then how can we commit a date?” However, the sales people will commit a date with the customer and

that is the struggle area. So, basically, all the development team needs to have the agile mindset. It is good to involve the customer and to let them know about the procedures as well. Unless they do it, it is very difficult to implement agile.

8. **How do managers and their respective teams combine agile and lean production principles in concrete projects?**
    - **Scrum, Kanban or Scrumban?**
    - **Challenges?**

Let us talk about lean first. The lean principle is “to remove waste”. So in the working processes, it is the low value activities that should be removed in order to only do the things that produce high value to the customer. This is realized by looking at the each activity of the project and doing a value stream mapping. A value stream mapping consists on assigning a value to each activity and identifying those with the lowest value in order to remove them. In this way, the process becomes lean (smart, not a bulky process). These lean principles can be applied in project management (whether in traditional or agile methods).

This might encounter its own challenges such as identifying what are the activities (those less valuable steps of the process) that need to be removed. Normally, people are reluctant and do not want to remove anything.

Getting lean is very hard. You have to measure and improve. People do not feel very comfortable with measuring. That is a big challenge when integrating lean.

Scrum is very useful for new developments and Kanban is very useful for maintenance. The ideal is to have different people for different tasks: some of them working on new features’ development and other people working to fix defects. It takes time to fix the bugs and people have to make a good analysis before starting the work. This process takes its own time and cannot be done by the same people that work on the new features, so using Scrum for one purpose and Kanban for the other would work great.

9. **What do managers do for setting and maintaining the right connections among people-processes/principles-technology?**
    a. **What do managers do to make developers committed with their job?**

Managers should set up the processes and then encourage teams to follow those processes, seeing what kind of issues the team is facing when following those processes, correcting and supporting the team, etc.

10. **Do you have any suggestions on how to improve productivity (in software industry) based on your personal working experience?**
    - Measure the productivity
    - Find the reasons why it is like that – finding the root causes
    - Fix the root causes

G. Lian Zhang

1. **How does your organization measure the productivity of the employees and their working processes?**
    a. **How does your organization measure the progress when aiming for effectivity and efficiency?**
    b. **(if they do not have any specific methods to measure productivity) what kind of experiences would help project manager to interpret efficiency and effectivity?**

In my organization, we have a process to set several milestone for a work package (or product development life cycle). preparation, formal start, commitment made, iterations (1-N), done and close. For each stage, there are some checklist to make sure all the criterion are fulfilled.

2. **Which project management factors make a difference in the development speed and productivity?**
    a. **What management actions (or "non" actions) would hamper software development speed?**
    b. **What management actions support higher productivity?**

To the speed and productivity, the most important management factors in my opinion is the quality. What kind of quality level decide if you can have a fast speed and high productivity with acceptable quality or normal even slow speed with high quality. So to me, the speed is important but a high quality productivity is more important.

3. **What is the effect of team size on productivity?**
    a. **Does team size affect the communication and collaboration among its members?**
    b. **Is communication and collaboration of the team members a crucial factor to achieve high productivity?**

According to the agile principle, 5 to 8 is OK for a effective team size. However it depends on what tasks or feature the team takes. If a team is more than 8, then the management and coordination cost will become a bit high. So to my preference, 8 is the most effective size.

4. **Are the projects with faster development cycles more productive?**
    a. **(If not) what are the decisive factors?**

It is a bit similar to the question 2, faster speed cannot promise better quality. And regarding the development cycle, as we are using Scrum, usually we choose 3 weeks as a iteration and the life cycle depends on how big this feature is.

5. **How can agile project management methodologies improve processes and methods?**
    a. **Consider a recently completed project and estimate its duration if it had been undertaken five years earlier and with traditional software development methods. How have agile methodologies contributed to the product development speed?**

The old way of development in most case is water-fall model. Requirement -> design -> implementation -> testing -> delivery and O&M. The biggest problem for this is when the requirements change if the team has already enter the testing period, all the previous phases will be impacted, sometimes, it is very costly and time consuming. While with the agile way of working, the whole feature is spitted into several iteration or sprints. All these development phases will be happened in each sprint and customers can get the demo or prototype very fast, so the agile helps the software development a lot in the changes handling. And also by doing this way, for each iteration, team can do the retrospective and get improved in next iteration. So the biggest benefits for the agile is changes handling, faster feedback, continuing improvement and effective management.

6. **Which practices are considered the best to be implemented when developing software?**
    a. **Technological aspect**
    b. **Human aspect**

No best practices.

7. **What are some approaches to moving an entire development team to Agile?**
    a. **Which are some challenges faced during this process?**
    b. **Do all members of development teams need to be on agile methods?**
    c. **What is the best practice to move the required people to agile?**
    d. **What are the challenges in doing so?**

Give training to the whole team and also the training to the person who will act as Scrum Master. After that, start a small feature development to let the team try with agile. It's

better to send a coach to the team and do the agile daily activities together to point out the problem and places need to be improved. The more team try, the more benefits team will find comparing to the old way of development. The biggest challenge to apply this new way of working is changing people's mind. If you have get used to and also play quite well in the old way, it's takes time to make people accept new things. Comparing to the old way, people will be more involved in the agile, faster reaction, daily status check in the daily meeting, etc. will bring more stress and pressure to some developers. Short pain is inevitable but when the team become a experienced agile team, they will definitely love it and use it.

8. **How do managers and their respective teams combine agile and lean production principles in concrete projects?**
    a. **Scrum, Kanban or Scrumban?**
    b. **Challenges?**

There is no rule say in which way should the team run the agile. Agile is just a principle to guide us to follow. Team can choose whatever way of agile to use depends on the situation. For instance, to a support team which is handling a lot of customer ticket report, the Kanban is a better way than a Scrum, but of course, they can still use the daily meeting and retrospective meeting practice from Scrum. To a Scrum team which gets in a new developer, pair-programming from XP can be used to speed up the competence building for this new guy. So agile is quite flexible. Even sometimes, team can try with some practice from different agile way of working to see if it helps or not. We also learn and used some concept of Lean to our work.

9. **What do managers do for setting and maintaining the right connections among people-processes/principles-technology?**
    a. **What do managers do to make developers committed with their job?**

People is the key, principles is a guideline, process is a proper way to ensure people can follow the principles. If you want people to follow a principle, use a proper and step-by-step way to stimulate people's interest and show them the benefits is important. Sometime a practice or show a demo from other organization. When people show interest and want to try with this principle, set up some rules to make sure people can follow up, this is process.

10. **Do you have any suggestions on how to improve productivity (in software industry) based on your personal working experience?**

    - Use the agile to improve the team working efficiency and improve the team velocity continuously.
    - Make development & testing as much automatic as possible.
    - Keep on focusing on people's training and competence build. Keep people learning and improving.

H. Paul Oldfield

1. **How does your organization measure the productivity of the employees and their working processes?**
    a. **How does your organization measure the progress when aiming for effectivity and efficiency?**
    b. **(if they do not have any specific methods to measure productivity) what kind of experiences would help project manager to interpret efficiency and effectivity?**

I shall answer for various organizations I have been associated with. The better responses consider what it means to be effective and efficient, and thus base their measures on the views of the customer. A good all-round approach to this was Capgemini's "OTACE" measure; "On Time, Above Customer Expectation" (the measure has since changed). The idea is that each project or other piece of work needs to get a deep understanding of what the customer expects, and to keep track of this as it changes with time. The "OTACE" score is in the customer's gift; we give the guidelines to say what the individual scores mean, they decide where we fit on the scale for the work we delivered to them.

A related measure that is often spoken about but has never been used by the organizations I've been associated with is "Net Promoter Score".

Below and contributing to the customer view we might measure efficiency directly. Aside from the normal financial 'profitability' measures that all organizations of any size will have, if only to keep their shareholders happy, one measure that the developers get involved with is **Function Point Counts**. Function Points are an independent measure of "Size" and can thus be used as benchmarks between teams, and have also been used to establish the basis of a fair cost. What effort does it take to produce a Function Point? Well that varies; Data-Rich systems give easy FPs, Control-Rich systems can involve considerable work just to produce one FP. Thus, there is generally a 'weighting' system applied to the context of the project that gives a mapping to how much effort it should take to produce a Function Point in that context. Of course we measure larger chunks; each chunk will be several to many FPs.

While there have been measures proposed to measure effectiveness directly, I don't recall any being applied by any of the organizations I have worked with.

My opinion is that it is what the customer thinks that really matters in this respect; happy customers tend to result in more business; both from them and via referrals and

testimonials. My opinion is that measuring progress is best done through early and frequent delivery, and by gathering feedback and responding to it so each subsequent delivery grows ever closer to the customer's idea of "Great".

2. **Which project management factors make a difference in the development speed and productivity?**
    a. **What management actions (or “non” actions) would hamper software development speed?**
    b. **What management actions support higher productivity?**

This rather depends on what you call "project management factors". I like to think of Project Management in an agile world as chiefly giving the team the work, the priorities and the resources, and taking away the impediments and the outputs.
So, I think that the problem areas are in giving the resources and taking away the impediments; it is important too for the management not to *become* an impediment. I like the "Lean" concept of a Manager - one who manages the process, who looks for flaws in the flow of work and smooths them out, who looks to decrease the cycle time and to remove waste. Instead, what we commonly get is a manager who manages the dependencies... a good manager would eliminate as many dependencies as possible. Attitude is important - the team are the ones doing productive work; the Manager is the person trying to ensure nothing gets in their way, that they have a steady and smooth supply of work and whatever they need to get the work done. One common problem I see, and this is "opinion" based on observation, is that Managers overvalue Managers.

3. **What is the effect of team size on productivity?**
    a. **Does team size affect the communication and collaboration among its members?**
    b. **Is communication and collaboration of the team members a crucial factor to achieve high productivity?**

There's that often quoted "magic 7 plus or minus 2". That applies quite well with respect to ideal team size. I shall touch on the problems with smaller teams and the problems with larger teams - and try to be concise; though there is a lot that could be said.

You need to have, either on the team or readily available to the team, all the skills that are needed to get the job done. For relatively simple jobs it might be relatively easy to get one or two people who between them have all the skills needed. As the job becomes more complex and requires more obscure skills, it is likely that you will need several people before you get a group who cover enough of the skills between them to form a viable team for the job.

The drive toward having more than the 'ideal' number of people is the volume of work that needs to be done. There are three main approaches to this problem. Of the three, in many respects the approach taken by "*37 Signals*" is the most interesting; they claim that *"*any* project can be undertaken by a small team of people; with more work you need better people"*. Of the better people I have conversed with, all claim to have worked on teams that are about 20 times more productive than the average team. This probably puts a limit on how much even the best of teams can get done, but certainly it is an approach that should not be dismissed out of hand as exaggeration.

The traditional approach is to have larger teams. The problem here is one of communication - with a large team you can't all be in continuous communication with each other and the work needs to be steered through parts of the team. The agile approach - or at least the more common ones of the agile approaches - acknowledge this and would run a large project with a collection of small teams. Now the problem is of how to divide the work between teams - this becomes a job of organizing dependencies. You can maximise dependencies by having functionally specialized teams; analysts handing over to designers handing over to coders handing over to testers. This is really inefficient and gives Managers a lot of dependencies to manage... Managers often favour this approach because they *understand* how to manage dependencies. You could split the system into architectural components and have a team working on each component - known as Component based Development. Again this is very inefficient; any single feature will likely require enhancement of several components and one needs to ensure that the delivery of the various parts of a feature are co-ordinated - again you have many dependencies to manage; again managers tend to like this. You can split the work into Feature Teams - one team per feature, working on the complete feature no matter what parts of the system it touches. This approach is favoured by agilists; it minimises dependencies; the down-side is that it maximises conflicts - conflicting updates to the same component. For a set of teams to work efficiently this way they need good inter-team

communications and good technical skills for managing conflicts - or more generally for "Embracing Change".

4. **Are the projects with faster development cycles more productive?**
    a. **(If not) what are the decisive factors?**

In my opinion there are two good aspects (among many others) that are the prime reasons for wanting short cycle times; neither of them are specifically and directly concerned with productivity, though in the nature of things, all things are connected to some degree.
The first and possibly most important gain with short cycle times, and I am talking about gross reduction in cycle times say from a year down to a month, is early return on investment. If you build the most valuable functionality first, and release it as soon as possible, that functionality can be earning money as soon as possible. Often the value of this first release is enough to offset the cost of the rest of the project so in such a case you have recouped your investment in a month rather than in a year.
The second gain with short cycle times (and again I am talking about gross reduction in cycle times) is early feedback. So, you released something after a month. The users spot a problem; you get the problem, and the *mistaken understanding* that led to the problem, fixed about a month into the project. For the whole of the rest of the project you are not introducing more problems based on the mistaken understanding.
Now once we've made the gross reduction in cycle time by releasing smaller increments, let us consider the fine adjustments of cycle time - releasing the same-sized increments in a shorter time. Now here there is a direct relation to productivity. By shortening the cycle time without reducing the size of the increment you are doing the same amount of effective work in a shorter time, thus by definition, as it were, increasing productivity.
Generally this shortening of the cycle time is achieved by identifying and removing waste, and you will find Mary Poppendieck's books on Lean Software Development a good source of information on that topic; for a free online introduction look up the "Lean Primer" by Craig Larman and Bas Vodde. If I recall correctly it touches on the topic, but it is certainly in any case useful background reading.
And on that topic I shall throw in a quotation from the guru of Toyota's Lean programme: "*The biggest waste of all, completely overshadowing all others, is the waste of recreating knowledge which we already possessed*" -- Taiichi Ohno

5. **How can agile project management methodologies improve processes and methods?**

a. Consider a recently completed project and estimate its duration if it had been undertaken five years earlier and with traditional software development methods. How have agile methodologies contributed to the product development speed?

As far as I can see (i.e. Opinion), traditional Management practices are based on "Best Practice" from Production environments. In production environments, you build many instances of the same thing over many times. You want to do the same thing in as economical a way as possible. Variation is not good. If instead we look at Development processes, it is the variation that adds all the value; there is no value in developing something exactly like what you have already developed. Traditional schools of management would seem to fail to make the distinction between the three main types of process that I identify: Production, Development and Service.

This problem is endemic and far-reaching. HR departments are possibly one of the worst areas - "Best Practice" in the HR world stems from the time we were a manufacturing nation. It is set up to manage people in a Production process. It puts people into pigeon-holes of skill, making specialists when we want generalizing specialists and a flexible workforce. It evaluates and rewards people as individuals, rewarding competitive behaviour when we want collaborative behaviour. It is geared to the "Simple" work environment of Production, when we in development are working in either "Complicated" or "Complex", occasionally "Chaotic" environments (as identified by the "Cynefin Framework").

Agile management comes in two forms. One form seeks to isolate the development team and allow agile working to be encapsulated within, and insulated from, a less than agile organization. Increasingly the whole organization is taking the Second form, as the whole organization becomes Lean or Agile; becomes a "Learning Organization". Here the development team is better integrated into the business process as a whole, and indeed may cease to exist as a "development team", becoming instead a capability of the "business team" as a whole.

Trying to compare Traditional and Agile directly is fraught with problems because one never does the same work twice so there can be no direct comparisons. Further, part of "being agile" would cause us to trim unnecessary items from the scope, while "traditional" approaches want to put as much as reasonably possible into the initial scope on the grounds that adding them later would be more expensive... so "like for like" comparisons just aren't

"like for like". As an example, back in 1997-9 I was working on a project in an agile fashion that we estimated in the range of 25-35 man-years. At the same time, "similar" projects were running along traditional lines, with estimates in the range 200-400 man-years. Yet, it would be very hard to pin down how much of the difference was owing to better productivity and how much was owing to better trimming of scope.
Again, it would be hard to say how much of this improvement would be down to better project management, and how much would be due to project management getting out of the way and letting the productive people get the work done.

6. **Which practices are considered the best to be implemented when developing software?**
    - **Technological aspect**
    - **Human aspect**

In some respects this is the easiest part to answer, in others it's the hardest.

First, there are NO universally good practices. Okay that is an opinion, but it is one shared by the original attendees of the Agile Alliance at Snowbird, 2001. They could not agree on any single practice that was best in all contexts, and they were probably right not to agree. However, they did agree on a set of values and a set of principles, and those are both (in theory) applicable in all contexts.

Well, I was about to go on and list some practices that all common agile approaches have, but I cannot think of any. There are certainly *types* of practice that they all have; there are properties of a process that all have, let us say. Properties such as that the process covers "knowing what it means to be on track". It covers "Knowing when you have gone off track". It covers "Continuous Improvement" one way or another, and in that respect, it covers "adapting to the context". It covers "Getting rapid feedback". It covers "Giving the customer early opportunity to accept delivery". It covers "Knowledge transfer" and "Getting the right skills engaged in the work". It covers "teamwork and team communication" It covers "Making sure that technical debt doesn't build up". Nevertheless, each process might have variations on the practices it selects to achieve these "properties" of the process.

Perhaps the best reference I can give here is some stuff I wrote over 10 years ago:
http://www.aptprocess.com/whitepapers/risk/RiskToPatternTable.htm

7. **What are some approaches to moving an entire development team to Agile?**
    a. **Which are some challenges faced during this process?**
    b. **Do all members of development teams need to be on agile methods?**
    c. **What is the best practice to move the required people to agile?**
    d. **What are the challenges in doing so?**

Here's an area where I've given generic advice before - with the usual caveat that people need to assess for themselves how good it would be in their specific context.
I identify two broad approaches - let us give them descriptive names of "Big Bang" and "Baby Steps". The key discriminant in choosing which of the approaches you should go for, is the size of "chunk" of work you release, in your current process that you are moving away from - that is; "What are your team accustomed to?" Now if the team are accustomed to delivering in small chunks, ideally considerably less than two months work in each, then I recommend the "Baby Steps" approach (I'll touch on "Why" later). If they are working Waterfall, and certainly if their delivery cycle is 2 months or longer, I recommend the "Big Bang" approach.
In this respect of the question, it is irrelevant whether you are a manager or a development team member, the reasoning applies to both. I might need to stress to specific managers that it *does* apply equally to them.
So, Why "Big Bang"? Where a team is accustomed to Waterfall development there tends to be a whole culture associated with it, and in effect to get Agile to work, we need to break that culture, or at minimum we need to break the culture for the team and its immediate 'manager'. In this respect "Scrum" is a good choice for several reasons, and that is probably why Scrum is so popular as an Agile approach. First, it has several fixed rules - if you do not follow the rules, you are not doing Scrum. To apply the rules you *must* rid yourself of a lot of the Waterfall thinking. In effect, you start your process with more or less a clean sheet, and then build it up again from fragments of what you know and from new practices that are known to be compatible with Scrum. Second, in the Scrum Master and Product Owner roles it goes a long way toward isolating the development team from the bad effects of the non-agile corporate culture, so you get a lot more time to try and break or change the corporate culture. That will not be an easy or quick task.
On the other hand, if you are already delivering your work in small pieces, ideally no more than a few days worth at a time but we might go up as far as 2 month chunks, then your team already knows, in some form, a lot about what is needed to be agile. Here you can

proceed incrementally to introduce the necessary changes to make the team more agile, a bit at a time. The culture can change at its own pace, provided that pace is not *too* slow. Kanban Development is a reasonable option to choose when already delivering small pieces of work; Scrum is probably better for the larger end of the scale.
No matter what approach you take, it is essential to have somebody who is reasonably conversant with agility, who can spot when the process of agile adoption starts to go off track, who can nudge it back on track. If you do not have such a person internally, I would advise hiring a reputable Agile Coach.
It is particularly important that it is realised that *anyone* who directly interacts with the team will probably need to change their thinking. Note that I talk about changing *thinking* rather than merely changing what they do. Agile asks folk continually to improve; continually to adapt to circumstances as they change. We want each individual who needs to adapt to understand the thinking underpinning Agile, so they can make sensible choices each time they need to make a choice.
All members of the development team need to work as a team. You recall I mentioned a property of the process was to get the right expertise engaged? One of the nearly universal "practices" of agility is the "*Self-Organizing Team*" - the team who are given, or take, the work in a big chunk and decide among themselves how it gets done, who does what part of the work. Now it is theoretically possible for a self-organizing team to adapt to accommodate a member who needs to have work allocated rather than choose for himself / herself, but in general there are no exceptions... and in my opinion if you are part of a self-organizing team then you are already, by definition, part-way toward being agile.
Therefore, at minimum all members of the development team need to be included. I also already mentioned that anyone who interacts directly with the team needs to be included.
"Best Practice" to move... Well I covered this last time, so here I will just have a little rant about the use of the term "Best Practice". I see the term "Best Practice" as meaning "Do this. Don't waste your time thinking about it". That is a very un-agile way of thinking; people should always understand why they do what they do. Waterfall itself is a fine example of the perils of "Best Practice" ...(opinion)... it was a sensible way of working back when CPU time was really expensive and compilers were really slow. You wanted to do all you could to ensure the program would compile correctly and work correctly, before you sent it to be compiled. Yet people forgot why they were doing this; it got labelled "Best Practice", and you still have people thinking Waterfall is best practice long after the reasons for it being 'Good' have disappeared.

The challenges? Many. Very many. People see it as taking away their power; people see it as undermining the knowledge they have built up over the years, people don't like to have to think, people think it doesn't apply to them, people find it really hard to change even when they want to, because they have forgotten *why* they do things the way they do, so are afraid of what will break if they change... that list covers just some of the major problems, and each of these will manifest in many ways. Then you get the keen ones - the ones who want to throw away all they know and get stuck in to a "new and better way of working" without understanding the problems this will bring, not just for them but for the project as a whole. When you are *already* agile you can deal with each of these problems as they arise, but while you are still learning, while changes in approach and tweaks to the process are still painfully slow and difficult, each of these problems has the potential to become big and serious.

Well, I have taken a viewpoint on this section; other viewpoints exist and they have important things to say. If they did not, there would be far fewer books on agility. I hope that other people will give you other viewpoints; I have my own opinion on what matters most.

8. **How do managers and their respective teams combine agile and lean production principles in concrete projects?**
    a. **Scrum, Kanban or Scrumban?**
    b. **Challenges?**

Interesting question. I would suggest, based on observation, that there aren't many folk knowingly attempting to combine agile and lean principles, but that there is a fair degree of combination already present in the "received wisdom" that they apply in their projects. For myself, I think of what I do as having a big toolbox and selecting the right tool for the job, no matter where and why it originated.

I wrote an article for the Capgemini internal Software Engineering magazine, while there were drives running in parallel to become more agile and to become more lean. To summarize, the two approaches have a great synergy. As I said in a bit of publicity for Alan Shalloway's book on the topic: "For a good few years, when asking why Agile approaches work, we got the response 'It's empirical. We tried things and kept the ones that worked'. Now people have applied theory from the Lean body of knowledge, and it tells us

why Agile approaches work. Using this theory, we can make well-reasoned choices about what changes to our ways of working would be improvements, overall".
While Lean and Agile have a great synergy, there are two specific areas where they differ, and I have to say that in the *ideal* world, "Lean" has the better approach in both cases... but few of us live in the ideal world. Lean asks us to consider the whole process, "Concept to Cash" as Mary Poppendieck suggested. Agile separates out the systems development part, and in some cases deliberately isolates the development team from the people working on the rest of the process. The other major point of difference is that Lean expects the Manager to be an expert in the process; to manage the work by managing the process. Agile has an unwritten assumption (nevertheless often true) that the manager will be ignorant of the nature of the development process and managing the process is best left to the development team.
In short, "Agile" in its more common forms is designed to allow agile working where the organization as a whole is NOT lean or agile; Lean is designed to encompass the whole organization - and where it does, a fair proportion of the common 'agile management' aspects of agility are out of place.
Again, a lot of this is 'opinion' but I believe it to be a useful and instructive viewpoint.

9. **What do managers do for setting and maintaining the right connections among people-processes/principles-technology?**
    a. **What do managers do to make developers committed with their job?**

I suppose in the "Lean" world this is indeed part of the Manager's job. However, in the Agile world this would be beyond the ability of the average manager. Let us start considering the team first, then the externals, and let us start from Principles, through processes and People.
Principles are things one believes to be the truth; they are about your world view. Changing people's beliefs can be done in most cases, provided the people don't have too much invested in the old beliefs that you're trying to override. However it isn't easy; you might need to work on two or maybe all three of the rational, emotional and political levels for any one person, and cover both the thinking and also tangible demonstration... as they say, "Seeing is Believing".. but even that isn't a guaranteed success. What works for one person may not work for another.

Practices and Processes - here what matters is what is appropriate to the context. An appropriate process meets three needs; first, it must be Adequate - it must deliver the outcomes that are needed, in the context that it will be enacted. It secondly must be Attainable - within the capability of the people you have available to enact the process, or to learn in a timely manner to enact the process. Thirdly, within the constraints set by the first two, it should be as economical as you can make it.
People and communication - within the team it is best to work as a "Self-Organizing Team" - this engenders a context where the team members know what communication ought to happen when it ought to happen, and they will ensure that it happens. So, that's an ideal; even the best teams occasionally miscommunicate. However, this form of communication network is far better than having a static, formal set of communication channels - it readily adapts to the immediate needs of the situation.
Going outside the team, while one can hope that other people have the same principles it would be unwise to assume they do; and interaction will be a compromise - to find a form of interaction that both parties can believe in. Practices and processes crossing team boundaries are going to be more static and not so readily adaptable. This gives reliability and predictability for a relatively small amount of effort, but will not adapt to changing circumstances very readily. Static, formal sets of communication channels are the norm at the outset, but these can grow into more flexible channels as the participants get to know each others needs, interests, preferences etc.
The key component in all of these is organic growth to adapt to changing circumstances. This is far more successful when the people who need to communicate adapt locally and collaboratively, rather than waiting for a manager to detect the need and tell them what changes ought to take place.

10. Do you have any suggestions on how to improve productivity (in software industry) based on your personal working experience?

Well of course we work, as teams, on that all the time. How about across the industry? I think the single biggest change for the better would be to have a tier or two of management who really understood development process and how it differs from production process. I do not know what they teach in the Management colleges these days, but the colleges seem to turn out managers who do not understand what a Development process is and how it differs from Production process. A few weeks work on, say, the Cynefin Framework and

its implications, or perhaps Don Rienertson's work, particularly "Managing the Design Factory" (for those colleges who like a good, scientific handling of the topic) would be a good start. One of the biggest problems trying to get Lean and Agile ways of working in place is the Manager who thinks he knows better. He has no real reason to listen to the staff when they seem to be telling him that a large swathe of what he believes is wrong, or at least not applicable in this context. After all, *he* has a degree in the subject, and they are not even managers so cannot understand the problem.

It is this shortage of understanding by Management that I would say is the single biggest impediment to improvement in the industry as a whole, because it has everybody pulling in different directions. A thoughtless few minutes work by a manager can undo weeks of hard work toward an improvement by a team.

An example? I recall setting up a process improvement process for one organization. The managers liked the idea and took over the project. The resultant process was designed to manage single improvements through a series of stage gates that were tied to the financing of the work. It was a process totally unsuited to managing process improvements because it completely ignored the engineering process that would be necessary to ensure a change would be an improvement; it completely ignored the fact that improvements were a "Many-to-many" relationship with the inputs to the process, and it completely ignored the reality that to get an improvement to stick you need to involve and get buy-in at several stages in the process from the people who would be working in a different way as a result of the change. Luckily, it also completely ignored that many changes need very little finance, so we just made a series of small changes and ignored their process. The managers saw improvement as something designed by experts and imposed on the team (as you might if designing a *Production* process for low-skilled workers, maybe).

I. Steve Fenton

1. **How does your organization measure the productivity of the employees and their working processes?**
    a. **How does your organization measure the progress when aiming for effectivity and efficiency?**
    b. **(if they do not have any specific methods to measure productivity) what kind of experiences would help project manager to interpret efficiency and effectivity?**

We want to be effective, possibly more than we want to be efficient - I'll explain this comment first. Being efficient is always the most effective way to work, it has undertones of high resource allocation, whereas we are aiming for good flow with a reasonable amount of down time. This is probably semantic really, so I'll now answer the question properly.

We find the best measure for effectiveness in lead and cycle times. We try to capture the length of time it takes a feature to transition between states - especially the time from "in progress" to "done". This gives a better indication of effectiveness than trying to count bugs, or hours, or comparing estimates to actuals as these are all meaningless outside of their specific context. Shorter cycle times are better. Long cycle times are worth investigating to see if the process needs fixing.

Another important metric is subjective wellbeing. The mood of the team indicates problems long before statistics and metrics. If the mood drops and nothing is done to improve it, the cycle times will start to stretch and the pace of the team will slow. This can only be measured by observation - but a happy team definitely produces better work and faster than an unhappy team. You need to sit near the team to notice these changes.

2. **Which project management factors make a difference in the development speed and productivity?**
    a. **What management actions (or “non” actions) would hamper software development speed?**
    b. **What management actions support higher productivity?**

The main role for project managers or managers is to give the team context and assist with blockages. In terms of context, if the team knows the product / business strategy and the

features that need to be developed as part of that, you can trust them to do the right thing. The more control you assert, the more you are relied on to make lower importance decisions. Management effort is best spent making sure the right people are talking to each other and helping the team to follow their own process.

3. **What is the effect of team size on productivity?**
    a. **Does team size affect the communication and collaboration among its members?**
    b. **Is communication and collaboration of the team members a crucial factor to achieve high productivity?**

We recently added two new team members. Initially, this seriously reduced the flow of work. The long-term effect was an overall increase. Whenever you change the line-up you will take a hit in this respect whether you are adding or removing team members. It is hard to give consistent results with less than 4 people and different tactics are required when the team gets bigger than 30 people.

4. **Are the projects with faster development cycles more productive?**
    - **(If not) what are the decisive factors?**

If you take a two-week cycle time as a theoretical base point, cycle times that are longer can make planning quite hard and you can travel much further in the wrong direction if nobody is checking the map for a long time. Two weeks is just about right - it is enough time to complete a substantial set of features that is worth releasing, but not long enough to make many mistakes. If you want to iterate faster than this you need to start dropping out some of the process artefacts (i.e. if you wanted to run weekly cycles, you wouldn't want to spend half a day planning, an hour doing a retrospective and an hour doing a demo). You can choose to release faster or slower than your development cycles.

5. **How can agile project management methodologies improve processes and methods?**
    a. **Consider a recently completed project and estimate its duration if it had been undertaken five years earlier and with traditional software development methods. How have agile methodologies contributed to the product development speed?**

If you were writing an application using a heavyweight process, you would need to document the features before you started programming. If this business analysis phase lasted for 3 months, an Agile team could have delivered 6 iterations of work on the most important features.
If the original requirements changed in the heavyweight process, you would only find out after the programming stage when people can see what has been produced, perhaps even after the testing stage. This will be very near to the end of the project and there may not be much time to incorporate the change. Agile is designed to adapt to change. The customer would see something after the first two weeks and could change their mind about how things should work based on what they see. There is typically a great deal of hostility between analysts, programmers and testers due to differing interpretations of the specifications.

Because you can see the product emerging in an Agile methodology, you can choose to release version "1" with less features than anticipated, which means the product can start earning money sooner. It also gives you the option to cease development sooner if you wish - or cancel sooner if it looks like the market has moved or the product may cost more than anticipated.

Agile tends to show the reality of the status of a project, whereas the status of a project in a heavyweight methodology is usually not really known. Heavyweight project plans are constantly massaged to match the delivery dates, which you cannot do in Agile (except by reducing the list of features).

It is hard to compare my work in Agile teams with my work in non-Agile teams. Anecdotally, I worked in a CMMI process for a Financial Company on a floor filled with developers, analysts and testers - somewhere between 60 and 100 people in total - and they delivered very little compared to a six-man Kanban team in a telecoms company. As an example, it took three months to deliver an integration to a third party in the CMMI process, but only two weeks to undertake a much more complex integration in the telecomms company.

A great many places get amazing results from Agile because what they have before is chaos, or a dysfunctional process - but Agile also attracts a different type of person, which is another reason it outperforms other processes.

6. **Which practices are considered the best to be implemented when developing software?**
    a. **Technological aspect**
    b. **Human aspect**

    Self-organizing teams are a fundamental must-have. It doesn't scale to have a manager making all the decisions and the longer you manage a team, the less able you are to make good decisions about the work the team are experts at. It also doesn't work to impose rules - it is much better to give direction and context and let the team work out how to achieve what you are asking of them.

    From a technology point of view, you need to be able to deploy fast, this means having automated builds, tests and packaging. This is especially important when you have a short cycle time. Taking a day to deploy new code is too slow when you need to get new features in front of testers and be finished in two weeks.

7. **What are some approaches to moving an entire development team to Agile?**
    - **Which are some challenges faced during this process?**
    - **Do all members of development teams need to be on agile methods?**
    - **What is the best practice to move the required people to agile?**
    - **What are the challenges in doing so?**

    The most important thing to do when moving to an Agile process is do it by the book until you are comfortably settled. It is too easy to dismiss the uncomfortable practices when you are new to a process and this leads to the process failing. Once you are running by-the-book you can begin to adapt the method to better suit the context you are working in. Most failures attributed to Agile are really attributable to not actually following the chosen method and not understanding the Agile principles.

8. **How do managers and their respective teams combine agile and lean production principles in concrete projects?**
    - **Scrum, Kanban or Scrumban?**
    - **Challenges?**

    Very often, teams will start with Scrum. If they decide to shorten cycle times, they may decide that Scrum has too many artefacts and head for Scrumban or Kanban. Ideally, the team needs to be scientific with process changes. All too often they make a change without

being prepared to revert it if it goes wrong. The retrospective should generate a small number of actionable changes along with how they will be measured.

9. **What do managers do for setting and maintaining the right connections among people-processes/principles-technology?**
    a. **What do managers do to make developers committed with their job?**

Make sure people are having conversations. Many Agile processes are designed to make awkward conversations happen sooner - but people don't like conflict so they will want to avoid these discussions. Questioning the flow of work, asking about the burn-down and otherwise encouraging people to have the courage to speak up early are a big help. It also helps a great deal to ensure there is sufficient space to visualise work and process. A white-board for the team to post up the new rules decided in the latest retrospective can help the team stick to their process.

10. **Do you have any suggestions on how to improve productivity (in software industry) based on your personal working experience?**

You need to put a value on any activity you ask people to participate in. Eliminate low-value activities in favour of high-value activities. Typically, the team will resist low-value activities and you need to listen for this - sometimes they will just have to put up with it, but very often they are right.

Waterfall success: I worked on a real-time trading system developed using waterfall that was a great success. Interestingly, there was a limit on the size of documents to keep them small and we worked in short iterations and the analysts sat with the programmers (and performed the testing).

Agile success: I have used Agile in several companies. Where the process is disciplined, the teams perform very well. This is irrespective of industry (I have seen it in financial services, e-commerce, telecoms and healthcare).

J. Tim Vermeerschen

1. **How does your organization measure the productivity of the employees and their working processes?**
    - **How does your organization measure the progress when aiming for effectivity and efficiency?**
    - **(if they do not have any specific methods to measure productivity) what kind of experiences would help project manager to interpret efficiency and effectivity?**

There is no real measurement with numbers. There are however 2 more informal ways that are applied:

- Looking at the end of each sprint which stories have been finished. However, the term “finished”/”done” currently doesn’t include the approval of the tester but rather an indication of the developer indicating that the development is finished and is in test.
- At the end of each release, which includes several sprints, there is a “lessons learned” session. In this session, the team members mention what they think is working and what they think is not working.

2. **Which project management factors make a difference in the development speed and productivity?**
    a. **What management actions (or “non” actions) would hamper software development speed?**
    b. **What management actions support higher productivity?**

Actions that I have seen slowing down the development speed are:

- Involving team members in activities other than the ones of the Scrum team and that were not taken into account for the sprint planning. It happens sometimes that some resources are not always available during the sprint even though they were planned for the entire sprint, hence the sprint objective cannot be reached.
- Wanting to go too quick is another factor, i.e. making stories that are not very clear or leave space for interpretation and starting to develop this. This leads to frequent re-works and even re-works of re-works. Sometimes gaps are only discovered during the testing but then are not considered part of the scope and leads to a new

story for a re-work. Some of those re-works sometimes require dropping the code and starting from scratch. Sometimes, it looks like the thinking is done too late and the purpose is not to deliver quality but just to deliver something quickly. Not only does this way of working slow down the development speed, it also de-motivates the developer. When such changes occur for parts that are so-called framework, there is also very often serious regression.

Another aspect that can slow down the development process is the dependency on external parties. In the project I am currently working for, a certain technology is used for the coding. Some bugs encountered in the system require a solution from this external party and it happens that the supplier doesn't consider it high priority while it is blocking our progress for some stories.

A lack of agreements, procedures, etc. also has had a negative impact so far. Team members don't always know exactly what to do and make assumptions without realizing it. The amazing part is that procedures, processes, etc. were defined throughout the years but once the "agile project" started, all those things were omitted.

Management actions that can support higher productivity include, according to me:

1) A good understanding of the methodology applied (e.g. Scrum)
2) Clear processes, structure, etc. This ensures that people within the team will work the same way. This is even more important I think if the project is so big that several teams need to work together.
3) Making sure quality is built in from the beginning, i.e. already during the definition of the stories. Having JIT requirements doesn't mean that those requirements should be vague and incomplete.
4) Having the testers involved soon enough. Whether it is in a traditional or agile environment, having testers e.g. review requirements has an added value.
5) Limit project risks. E.g. in case of an external party, making sure that there are clear agreements with that party.

3. What is the effect of team size on productivity?

- **Does team size affect the communication and collaboration among its members?**
- **Is communication and collaboration of the team members a crucial factor to achieve high productivity?**

Even though the team size shouldn't have any impact, it does. Smaller teams seem to be more easily self-regulating or self-managing. The bigger the teams, and even more when several teams exist aiming for the same goal, more structure and clear agreements seem necessary. A lot however also depends on other parameters. A team of 10 people hardly communicating could produce less or less quality than a team of 15 people communicating well and having clear agreements.

4. **Are the projects with faster development cycles more productive?**
    - **(If not) what are the decisive factors?**

Projects with faster development cycles are not necessarily more productive. I mainly think about projects in which the quality of the stories is not high enough. I also think about projects where the framework is badly put together. Some agile projects seem to postpone the thinking step until after the development. It sometimes might look like such projects quickly deliver working software. However, when after some sprints, as the software gets more complex, it sometimes results in the realization that wrong decisions were made, and un-doing this sometimes slows down a project. A possible side-effect is that teams keep on fine-tuning and re-factoring code instead of developing new functionality.

5. **How can agile project management methodologies improve processes and methods?**
    - **Consider a recently completed project and estimate its duration if it had been undertaken five years earlier and with traditional software development methods. How have agile methodologies contributed to the product development speed?**

One of the advantages of an agile project management methodology is that the processes, methods and way of working can be regularly questioned. It allows fine-tuning those processes, methods and way of working. It shouldn't however be an excuse to start a project without having decent processes and methods. The purpose shouldn't be to just start and see where you get.

6. **Which practices are considered the best to be implemented when developing software?**
    - **Technological aspect**
    - **Human aspect**

One of the criteria that I consider important is to have good agreements and clear processes for everybody, i.e. in order to make everybody work the same way. Tools should be easy to use.

To work effectively, I think that it is important to have clear requirements, i.e. know what is expected. One of the key factors is communication. Next to this, also having someone who is able to ask the right questions when determining requirements plays an important role.

7. **What are some approaches to moving an entire development team to Agile?**
    - **Which are some challenges faced during this process?**
    - **Do all members of development teams need to be on agile methods?**
    - **What is the best practice to move the required people to agile?**
    - **What are the challenges in doing so?**

The approach at the company where I work was to start applying an agile project management methodology, Scrum, for a new internal project. The purpose was to see whether it would be successful. So no big bang was applied.
One of the main challenges is/was the lack of thorough knowledge about Scrum by management and people taking the lead in the project (Scrum master and product owner). This had (and still has) as effect that the project is not fully approached in an agile way.
Another challenge is the fact that there are influences from outside the team. Examples are resources being removed from the project while unannounced, the sprint back log being modified while the sprint is ongoing, etc.
A challenge I faced myself was the one of integrating a tester in the project and the question what should be tested inside the sprint and what should be tested outside the sprint.
It also seemed difficult to get rid of the “that’s not my topic” attitude.
A last big challenge was the dropping of all agreements and processes for the project. It took a lot of time to have those processes defined and fine-tuned in the company and for the project they were simply ignored while they had proven to be useful.
I think that giving an introduction to the team members about the agile methodology that will be applied is useful. It helps to have everybody know how it works and what is expected from them in the applied methodology.

One of the main problems when moving people into an agile team is to have them work as a team for the same objective. Some people are more a team player than others. Also, some people seem to have it difficult to work with people with a different background, or at least to not consider them as colleagues but rather as "someone of the other team". I've e.g. still seen developers arguing with testers. Another example is where people do what they like to do and act on individual basis.

8. **How do managers and their respective teams combine agile and lean production principles in concrete projects?**
    - **Scrum, Kanban or Scrumban?**
    - **Challenges?**

In my company, there is an attempt to apply Scrum. In the tool used, some of Kanban is used for the visualization of the status of stories.

9. **What do managers do for setting and maintaining the right connections among people-processes/principles-technology?**
    - **What do managers do to make developers committed with their job?**

Team members are made responsible of a story or other items. Also, team members are rewarded from time to time by management by e.g. going for lunch or bringing a box of sweets.
To make team members familiar with new concepts, handovers or trainings can be given. Where I work, this is usually done internally. Some people also learn new things by reading in their private time.

10. **Do you have any suggestions on how to improve productivity (in software industry) based on your personal working experience?**

- Don't drop existing procedures and agreements if they proved to be useful in the past. "Agile" doesn't mean "no procedures or processes".
- Try to make things right the first time. This starts by defining good stories and giving the right priority to a story. "Welcoming changing requirements" is not the same as "just do something as it can be changed afterwards". Also, a story should contain enough and correct information. When some information is not mentioned, it cannot be expected to have that developed.

- Involve testing soon. Have e.g. the quality of the stories for the next sprint checked upfront.
- Have people with the right skills and personality.

K. Johnny Hermann

1. **How does your organization measure the productivity of the employees and their working processes?**
    a. **How does your organization measure the progress when aiming for effectivity and efficiency?**
    b. **(if they do not have any specific methods to measure productivity) what kind of experiences would help project manager to interpret efficiency and effectivity?**

- Velocity in relative Story Points representing the new scope accomplished per two-week sprint/iteration. Long-term velocity should ideally stay steady or improve.
- Effort Creep in person-hours added to the sprint, after the sprint has already started.

2. **Which project management factors make a difference in the development speed and productivity?**

Generally, removing impediments from team members so that can execute as the professionals they are, and not be bogged down with unrelated overheads.

- **What management actions (or "non" actions) would hamper software development speed?**

- Interrupting developers.
- Scheduling additional meetings.
- Assigning work ("push"), instead of developers self-assigning ("pull")

- **What management actions support higher productivity?**

- Minimize all interruptions to software-developer mental flow.
- Remove / follow-up on impediments.

3. **What is the effect of team size on productivity?**
    - **Does team size affect the communication and collaboration among its members?**
    - **Is communication and collaboration of the team members a crucial factor to achieve high productivity?**

7 +/- 2 members is the recognized optimal team size, and we try to follow it. Anything larger will cause productivity to start to decrease due to overloaded communication-graph effects, i.e., O(n^2), and decreasing team cohesion.

4. **Are the projects with faster development cycles more productive?**
    - **(If not) what are the decisive factors?**

Yes up to a certain limit, which must be determine empirically for the particular team and project in question. The upper limit on "faster" (i.e., shorter) sprint cycles is determined largely by the coordination and transaction overhead of necessary meetings. A certain minimum amount of time is required for meetings to be useful (and to justify interrupting developers). But the longer those meetings are, the more percentage they take up of a shorter sprint.

5. **How can agile project management methodologies improve processes and methods?**
    - **Consider a recently completed project and estimate its duration if it had been undertaken five years earlier and with traditional software development methods. How have agile methodologies contributed to the product development speed?**

Team retrospectives via independent, anonymous, written feedback (hence unbiased) - such as modified Delphi methods - raise impediments to be resolved, and successes to be leveraged.
Agile does not necessarily increase development speed, rather increases reaction time to changes in requirements such that the product that gets developed is of high quality and still has relevant value aligned with customer / market needs.

6. **Which practices are considered the best to be implemented when developing software?**
    - **Technological aspect**
    - **Human aspect**
        - Regarding Agile: Retrospectives;
        - tight feedback loops between all team members;
        - time-boxing;

- limit work in progress (WIP).

7. **What are some approaches to moving an entire development team to Agile?**
    c. **Which are some challenges faced during this process?**
    d. **Do all members of development teams need to be on agile methods?**
    e. **What is the best practice to move the required people to agile?**
    f. **What are the challenges in doing so?**

The Lean approach (Kanban Method) to introducing Agile is least intrusive. It says to start where you are, and continuously improve based on team retrospectives towards more Agile practices.

It is recommended that you start Agile with a small isolated collocated team, and achieve some small, measurable successes. That evidence then promotes further expansion of Agile to other teams, potentially eventually the entire development team.

Challenges include psychological resistence to change; managerial expectations of command-and-control reporting; lip-service managerial support without follow-through; incompatibilities between Agile team and overall traditional org especially at touchpoints. Not all members need to be "on agile methods". But if not all Agile, the resulting friction (coordination & transaction overhead) can cause more cost than it is worth.

8. **How do managers and their respective teams combine agile and lean production principles in concrete projects?**
    - **Scrum, Kanban or Scrumban?**
    - **Challenges?**

Most contemporary Agile projects are hybrid Lean-Agile - some combination of Scrum & Kanban. Challenges depend on team maturity to maintain the discipline to follow-through on the related constraints of the methodology agreed.

9. **What do managers do for setting and maintaining the right connections among people-processes/principles-technology?**
    - **What do managers do to make developers committed with their job?**

Lean-Agile Lead (e.g., ScrumMaster) has a primary duty to facilitate the team in coming to concensus on processes, princicples, etc., then to remind and encourage the team to follow-through on their own agreements. A duty of all contemporary managers is to be servant-leaders in an upside-down org, supporting and nurturing their teams.

10. **Do you have any suggestions on how to improve productivity (in software industry) based on your personal working experience?**

    Improve the productivity of the team or organization, not the constituent individuals - since they are generally _not_ the problem (per the Fundamental Attribution Error). Local optimization usually leads to global de-optimization. Measure and optimize the aggregate complex adaptive system (CAS).

Appendix 4 – The online questionnaire and its results

Productivity improvement in Software Developing industry - Agile, Lean, Leagile paradigms

Master thesis questionnaire

* Required

1. **How long have you been working with software development project?** *
   Mark only one oval.
   - ◯ Less than 1 year
   - ◯ 1-5 years
   - ◯ More than 5 years

Software Development Productivity

2. **Which of the following options would be a good measurement for software developing productivity?** *
   Check all that apply.
   - ☐ Lines-of-code productivity
   - ☐ Function points productivity
   - ☐ Other: .....

3. **Which country have the projects taken place?**

4. **A faster development cycle affects the productivity of a project by making it....** *
   Mark only one oval.

	1	2	3	
Less productive	◯	◯	◯	More productive

Agile methodology

5. **Have you ever been involved in agile projects?** *
(if YES, go to the next question; otherwise go to the last one)
Mark only one oval.

- ◯ Yes
- ◯ No

6. **To what extent have agile methodologies improved software developing processes?**
Consider the effects that the usage of agile methodologies has on the productivity of the employees.
Mark only one oval.

	1	2	3	
No improvement at all	◯	◯	◯	Huge improvement

7. **Your role in a Scrum project:**
Mark only one oval.

- ◯ Scrum master
- ◯ Scrum coach
- ◯ Scrum practitioner

8. **Which of the following factors would mostly decrease your productivity?**
(To be answered by Scrum practitioners)
Check all that apply.

- ☐ A micro management style, where you have a list of tasks and you should follow it
- ☐ A macro management style that gives you allowance and independence in making decisions for your own work
- ☐ The cancellation of a sprint
- ☐ The lack of incentives (moral, monetary,etc.)
- ☐ Other: .....

9. **What do you do to increase the productivity of the team?**
(To be answered by Scrum masters)
Check all that apply.

- ☐ Provide sprint celebrations - free time activities
- ☐ Provide monetary incentives
- ☐ Providing direct contact with the customer
- ☐ Other: .....

- ◯ It is expensive to implement a new approach of developing software
- ◯ People are unwilling to share knowledge
- ◯ Establishing direct contact with the customer is difficult
- ◯ It is hard to coordinate the work of the people, especially in big projects compounded by several teams
- ◯ Other: _____

Lean philosophy - Leagility paradigm

11. **Do you have any practical experience in lean software development?** *
(if YES, go to the next question; otherwise go to the last one)
Mark only one oval.

- ◯ Yes
- ◯ No

12. **What do you think of a possible integration of agile and lean philosophies (leagile paradigm)?**
(if answered YES the previous question)
Mark only one oval.

- ◯ It is beneficial and worth to be implemented; the same people would focus both on agile and lean tasks.
- ◯ It is beneficial; It may be integrated in a project. but not within a single team as developers need to concentrate on what they are doing (whether lean or agile)
- ◯ It is time consuming and quite challenging therefore sticking to Scrum would be better.
- ◯ It is time consuming and quite challenging therefore sticking to Lean would be better.
- ◯ Other: _____

Suggestions based on experience

13. **Suggestions on how to improve productivity and increase business value in software developing industry:** *

How long have you been working with software development project?	Which of the following options would be a good measurement for software developing productivity?	A faster development cycle affects the productivity of a project by making it....	Have you ever been involved in agile projects?	To what extent have agile methodologies improved software developing processes?	Your role in a Scrum project:	Which of the following factors would mostly decrease your productivity?	What do you do to increase the productivity of the team?	The biggest challenge of implementing agile?	Do you have any practical experience in lean software development?	What do you think of a possible integration of agile and lean philosophies (leagile paradigm)?	Suggestions on how to improve productivity and increase business value in software developing industry:	Which country have the projects taken place?
More than 5 years	Value added	3	Yes		2 Scrum master		By preventing distractions and help in solving any problem/issue once it appears	It is hard to coordinate the work of the people, especially in big projects compounded by several teams	Yes	I would only say one thing and you should reach a conclusion: "You have to be LEAN to be AGILE". Lean helps is realizing the agile manifesto.	Sorry, no simple and straightforward answer. However, what I'd really emphasize is that whatever you do, if it does not add value from the customer's point of view, your work and productivity worths nothing!	Canada
More than 5 years	units-of-work, features with agregated value per iteration	1	Yes		2 Scrum practitioner	A micro management style, where you have a list of tasks and you should follow it, The lack of incentives (moral, monetary,etc.), frequent interruptions	Providing direct contact with the customer, intrinsic and extrinsic motivation should be combined along with autonomy	It is hard to coordinate the work of the people, especially in big projects compounded by several teams	Yes	It is beneficial; It may be integrated in a project. but not within a single team as developers need to concentrate on what they are doing (whether lean or agile)	there are many, many efforts in the academic and industry field to address this issue. I think, criticism on the methods and possibly combining them can increase both.	Brazil
More than 5 years	number of accepted features	2	Yes		2 Scrum master	A macro management style that gives you allowance and independence in making decisions for your own work	Providing direct contact with the customer	People are unwilling to share knowledge	Yes	It is beneficial and worth to be implemented; the same people would focus both on agile and lean tasks.	Exposure to the business value and technical aspects will improve it to a great extent	India
More than 5 years	Customer satisfaction	3	Yes		3 Scrum master	The cancellation of a sprint	Providing direct contact with the customer, Make the work meaningful (which is helped by meeting with customer so the value is seen)	Change of mindset, breaking things down small	No		Include User Experience in the evaluation of how well a product is doing and in evaluating what is produced.	USA, China, Poland, UK, Canada
More than 5 years	features released to production	3	Yes		3 Scrum master	unclear acceptance criteria	a strong product owner	support or lack of support from upper management	Yes	It is beneficial and worth to be implemented; the same people would focus both on agile and lean tasks.	create an opportunity backlog and a product backlog, have time to test out ideas and innovate.	USA
More than 5 years	Function points productivity with some measurement of bugs.	3	Yes		2 Scrum practitioner	The lack of incentives (moral, monetary,etc.), Not finalizing requirements for the sprint before the sprint starts.		Distributed teams	Yes	It is beneficial and worth to be implemented; the same people would focus both on agile and lean tasks.	Right set of tools and automation of mundane tasks. Solid regression and unit testing suites.	USA, INDIA

How long have you been working with software development project?	Which of the following options would be a good measurement for software developing productivity?	A faster development cycle effects the productivity of a project by making it...	Have you ever been involved in agile projects?	To what extent have agile methodologies improved software developing processes?	Your role in a Scrum project:	Which of the following factors would mostly decrease your productivity?	What do you do to increase the productivity of the team?	The biggest challenge of implementing agile?	Do you have any practical experience in lean software development?	What do you think of a possible integration of agile and lean philosophies (leagile paradigm)?	Suggestions on how to improve productivity and increase business value in software developing industry:	Which country have the projects taken place?
More than 5 years	working software	3	Yes	3	Scrum coach	A micro management style, where you have a list of tasks and you should follow it	Providing direct contact with the customer	Establishing direct contact with the customer is difficult	Yes	It is beneficial and worth to be implemented; the same people would focus both on agile and lean tasks.	Make sure that the development team understands the business value of what is delivered. Apply lean startup techniques to understand what the customer really wants and shorten the feedback as much as possible. Use impact mapping at product design Time based assignments with some target benefit attached to performance result will increase both the developers productivity and business value.	Ireland
More than 5 years	Function points productivity	2	Yes	3	Scrum practitioner	The cancellation of a sprint, The lack of incentives (moral, monetary, etc..)	Providing direct contact with the customer, Daily scrum	It is hard to coordinate the work of the people, especially in big projects compounded by several teams	No			Ethiopia
More than 5 years	Function points productivity	3	Yes	3	Scrum master			Establishing direct contact with the customer is difficult	No		Improve dev skills, agile project management	Switzerland

How long have you been working with software development project?	Which of the following options would be a good measurement for software developing productivity?	A faster development cycle affects the productivity of a project by making it....	Have you ever been involved in agile projects?	To what extent have agile methodologies improved software developing processes?	Your role in a Scrum project:	Which of the following factors would mostly decrease your productivity?	What do you do to increase the productivity of the team?	The biggest challenge of implementing agile?	Do you have any practical experience in lean software development?	What do you think of a possible integration of agile and lean philosophies (leagile paradigm)?	Suggestions on how to improve productivity and increase business value in software developing industry:	Which country have the projects taken place?
1-5 years	Fewer bugs / test cycle		3 Yes		2 Scrum master	A macro management style that gives you allowance and independence in making decisions for your own work, Fixed-time tasks	Provide sprint celebrations - free time activities, Giving free cake in retrospectives meetings, Giving freedom of choice, Motivating team members based on an individual level	People are unwilling to share knowledge	No		Make sure people know that knowledge sharing is not the same as loosing your knowledge and transferring it to the other person! This might reduce the fear level! Do not assume that every one is going to be motivated by the same free activitiy or piece of free cake! Encourage knowledge sharing and having actual cross functional teams. If you have very specialized tasks and people such that the nature of a cross functional and the nature of a team is lost and you are basically a group of specialized individuals working on the same project, consider stopping to use scrum and use Kanban instead. Keep the people away from the project who don't realise that quality is more important than delivery date. (economists and other non-engineers)	Sweden, China, Croatia, Canada
More than 5 years	Function points productivity		3 Yes		3 Scrum practitioner	A micro management style, where you have a list of tasks and you should follow it, The cancellation of a sprint			No			Sweden
1-5 years	Function points productivity		3 No			A micro management style, where you have a list of tasks and you should follow it, The lack of incentives (moral, monetary,etc.)			No		Finding balance between process oriented and what i call "free developing" in the team	Sweden