Productivity of agile teams: an empirical evaluation of factors and monitoring processes

Claudia de Oliveira Melo

Text submitted
to the
Institute of Mathematics and Statistics
of the
University of São Paulo
for the
Doctoral degree in Computer Science

Advisor: Professor Fabio Kon

This research was supported by FAPESP, Brazil, proc. 2009/10338-3,
CNPq, Brazil, proc. 76661/2010-2, and the Research Council of Norway, grant 202657.

São Paulo, May, 2015

Productivity of agile teams: an empirical evaluation of factors and monitoring processes

This version of the thesis contains corrections
and suggested changes by the Judging Committee
during the defense of the original version
of the work, held on 09/05/2013.
A copy of the original version is available
at the Institute of Mathematics and Statistics,
University of São Paulo.

Doctoral Thesis Committee:

- Prof. Dr. Arndt von Staa - PUC-Rio - Informatics Department
- Prof. Dr. Cleidson Ronald Botelho de Souza - UFPA and Vale Institute of Technology
- Prof. Dr. Fabio Kon - IME-USP - Department of Computer Science
- Prof. Dr. Marco Aurélio Gerosa - IME-USP - Department of Computer Science
- Prof. Dr. Sandra Fabbri - UFSCar - Department of Computer Science

Abstract

MELO, C. O. **Productivity of Agile Teams: An Empirical Evaluation of Factors and Monitoring Processes**. 2015. 195 f. Thesis (Doctoral) - Institute of Mathematics and Statistics, University of São Paulo, São Paulo, 2015.

Lower cost and shorter time-to-market expectations are the major drivers of software productivity improvements. To manage productivity effectively, it is important to identify the most relevant difficulties and develop strategies to cope with them. Agile methods, including Extreme Programming and Scrum, have evolved as approaches to simplify software development process, potentially leading to better productivity. They aim to shorten development time and handle the inevitable changes resulting from market dynamics.

Although the industry has extensively adopted agile methods, little research has empirically examined the software development agility construct regarding its dimensions, determinants, and effects on software development performance. Understanding this construct could help determine where to concentrate management efforts (and related financial resources) from a practical standpoint and where to focus research efforts from an academic perspective.

Considerable research has been directed at identifying factors that have a significant impact on software development productivity. In general, the studied productivity factors were related to product, personnel, project, process, or organizational issues. Continuously evaluating productivity factors is important, as factors may change under new software engineering practices. However, little research has investigated the major factors influencing agile team productivity.

The goal of this thesis was to explore productivity definitions, factors, and monitoring in *agile teams* and to improve the practice based on the collected evidence and gained knowledge. This thesis presents five novel contributions: C1 - Empirical verification of the importance of productivity for companies adopting agile methods and its perceived benefits; C2 - Rationale for the definition of productivity in the context of agile methods; C3 - Empirical verification of agile team productivity factors; C4 - A conceptual framework for agile team productivity factors and their impact; C5 - A team productivity monitoring process considering adaptability and an evaluation of the usefulness of agile team productivity metrics.

Keywords: Software productivity, agile software development, productivity factors, empirical software engineering, team performance.

Resumo

MELO, C. O. **Produtividade de times ágeis: uma avaliação experimental de fatores e processos de monitoramento**. 2015. 195 f. Tese (Doutorado) - Instituto de Matemática e Estatística, Universidade de São Paulo, São Paulo, 2015.

Menor custo e expectativa de menor *time-to-market* são os principais motivadores para melhorias de produtividade de software. Para gerir eficazmente a produtividade, é importante identificar as dificuldades mais relevantes e desenvolver estratégias para lidar com elas. Os métodos ágeis, incluindo Programação Extrema e Scrum, evoluíram como abordagens para simplificar o processo de desenvolvimento de software, potencialmente levando a uma melhor produtividade. Eles visam reduzir o tempo de desenvolvimento e lidar com as mudanças inevitáveis decorrentes da dinâmica do mercado.

Embora a indústria tenha adotado amplamente métodos ágeis, há pouco entendimento científico do construto *agilidade em desenvolvimento de software* em relação às suas dimensões, determinantes e efeitos sobre o desempenho no desenvolvimento de software. Compreender esse construto poderia ajudar a determinar onde concentrar os esforços de gestão (e recursos financeiros relacionados) de um ponto de vista prático, assim como onde concentrar os esforços de investigação de uma perspectiva científica.

Pesquisa considerável tem sido direcionada para identificar os fatores com impacto significativo na produtividade de desenvolvimento de software. Em geral, os fatores de produtividade estudados foram relacionadas ao produto, pessoas, projeto, processo ou questões organizacionais. Avaliar fatores de produtividade continuamente é importante, pois os fatores podem mudar quando novas práticas de engenharia de software são adotadas. No entanto, poucos estudos investigaram fatores influenciando a produtividade de times ágeis.

O objetivo desta tese é explorar definições, fatores e monitoramento de produtividade em *times ágeis* e melhorar a prática baseada em evidência. Esta tese apresenta cinco novas contribuições: C1 - Verificação empírica da importância de produtividade para as empresas que adotam métodos ágeis e seus benefícios percebidos; C2 - Justificativa para a definição da produtividade no contexto de métodos ágeis; C3 - A verificação empírica de fatores de produtividade em times ágeis; C4 - Um arcabouço conceitual de fatores de produtividade em times ágeis e seu impacto; C5 - Um processo de acompanhamento de produtividade de times ágeis, considerando adaptabilidade e uma avaliação da utilidade de métricas de produtividade para esses times.

Palavras-chave: Produtividade de software, desenvolvimento ágil de software, fatores de produtividade, engenharia de software experimental, desempenho de time.

Acknowledgments

To my family, that encouraged, supported, understood, and loved me at every moment. In particular, my husband Rodrigo and my sister Luciana. None of this could have happened without them.

I am deeply grateful to Professor Fabio Kon for giving me the confidence to explore my research interests and the guidance to avoid getting lost in my exploration. Fabio supported all my initiatives in the last 3 years and 9 months, helping me to focus on things that truly matter.

My sincere gratitude to all colleagues from Agilcoop, CCSL-IME-USP, and SOMA (not mentioning all names because I do not want to forget anyone). You were supportive every time I needed feedback or insights in a timely manner. Special thanks to my USP friends Thiago Sousa, Eduardo Takeo, Celina Takemura, Marcelo Palin, Clodis Boscarioli, Vladimir Rocha, Nafiseh Shabib, Mohsen Anvaari for the friendship, patience, laughs, and an appreciation of the lifestyle that a Ph.D. candidate experiences. To my friend Alexandre Vidal, who gave me significant advice to start, keep, and finish my journey.

Thanks for the four companies that participated in this research engagement: without them, I could not have achieved the research goals. Thanks to all survey respondents from the Brazilian IT industry. I am grateful to Professor Reidar Conradi, Dr. Daniela Cruzes, and Dr. Tore Dybå for mentoring my journey during my research visit in Norway.

My gratitude to FAPESP, CNPq, and Research Council of Norway, whose financial support made this work possible.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Software productivity is an important concept discussed in projects and organizations. At the beginning, the debate was centered on how much efficient and scalable a team could be when developing systems. The information age, however, has introduced complexity issues back into the software development practice, in a market eager to highly customized products and services. Being productive is not only being efficient, but innovative, fast learning, and adaptive. Software productivity has been changing ever since, but research studies and even industry are still tied to those old productivity concepts.

This thesis aims at exploring software productivity in an up-to-date view of software development, that includes a productivity concept review, an examination of the importance of productivity for companies, an exploration of factors influencing productivity of adaptive teams, and an investigation of approaches to identify threats to productivity in agile teams.

This chapter briefly presents our problem outline and the research context. It also describes our research questions, research design, and the claimed contributions to the software productivity field after a four-year research engagement. Finally, we[1] summarize our contributions through a list of papers and reports presented in national and international conferences and journals.

1.1 Problem outline

Lower cost and shorter time-to-market expectations are the major drivers of software productivity improvements [Boehm, 1987, Trendowicz and Münch, 2009]. To manage productivity effectively, it is important to identify the most relevant difficulties and develop strategies to cope with them. Agile methods, including Extreme Programming [Beck and Andres, 2004] and Scrum [Schwaber and Beedle, 2001], have evolved as approaches to simplify the software development process, potentially leading to better productivity. They aim to shorten development time and handle the inevitable changes resulting from market dynamics [Karlström and Runeson, 2006, Pikkarainen *et al.*, 2008].

Although the industry has extensively adopted agile methods, little research has empirically examined the software development agility construct regarding its dimensions, determinants, and effects on software development performance [Balijepally *et al.*, 2009, Dybå *et al.*, 2007, Lee and Xia, 2010, Shull *et al.*, 2010, Sudhakar *et al.*, 2011]. Understanding the factors that affect productivity could help determine where to concentrate management efforts (and related financial resources) from a practical standpoint and where to focus research efforts from an academic perspective [Rasch and Tosi, 1992].

Considerable research has been directed at identifying factors that have a significant impact on software development productivity [Trendowicz and Münch, 2009, Wagner and Ruhe, 2008]. In general, the studied productivity factors were related to product (specific characterization of software), personnel (team member capabilities, experience, and motivation), project (management

[1] One note regarding the thesis writing is that I often use "we" to describe what was done in the research. Since the research is a collaborative process, both with my supervisor, and other researchers that provide valuable feedback, I adopted this style.

aspects, resource constraints), or process issues (software methods and tools). Continuously evaluating productivity factors is important, as factors may change under new software engineering practices [Petersen, 2011]. However, to the best of our knowledge, no study in the literature has investigated the major factors influencing *agile team* productivity.

In addition, empirical evidence points out the lack of studies on agile productivity measurement [Dybå and Dingsoyr, 2008, Petersen, 2011], as well as team productivity measurement [Pries-Heje and Commisso, 2010]. However, at the same time, there is an increasing interest on agile performance measurement systems, both in academia [Lohan *et al.*, 2010] and industry [Highsmith, 2006]. Highsmith [2006] argues that we must change the way we measure performance to build agile organizations.

Throughout the research, we have also observed the lack of well-established productivity definitions and metrics in agile teams in the Brazilian IT industry [Melo *et al.*, 2011]. Thus, empirical studies on productivity definitions and metrics on agile teams may answer questions on how to improve agile teams management, as well as how to monitor productivity in a highly adaptive scenario.

In summary, the overall problem addressed by this thesis is:

The understanding of software productivity is still tied to old concepts that are not linked to the complexity and innovation demanded from agile development teams. This gap hampers the industry ability to effectively manage software productivity, and prevent innovative research that contribute to the area.

1.2 Research questions

The **goal** of this research is to **explore software productivity in an up-to-date view of software development**, that includes a productivity concept review, an examination of the importance of productivity for companies, an exploration of factors influencing productivity of adaptive teams, and an investigation of approaches to identify threats to productivity in agile teams.

This goal can be better investigated through the following research questions:

- **RQ1. How important is productivity for companies adopting agile methods? How do they define productivity?**
- **RQ2: What factors impact agile team productivity and how is this impact from the team point of view? Which agile practices are perceived to impact on a given team's productivity?**
- **RQ3: How to monitor productivity factors, considering agility and adaptability? How do agile metrics support productivity monitoring?**

Figure 1.1 shows the relationship between the research questions and a high-level productivity research framework. The framework show many possible factors impacting agile team productivity, whose impact can be continuously evaluated through objective and subjective measurement. The next chapters detail the framework based on an extensive literature review. We will investigate many aspects of the framework, including agile team productivity definition and importance, productivity factors in an agile team environment, and also how to monitor possible factors in a project.

1.3 Research context

The research in this thesis uses quantitative and qualitative data from four companies (hereby called A, B, C, and D) using agile methods for at least two years. The companies profile are summarized as follows, and detailed in the next chapters:

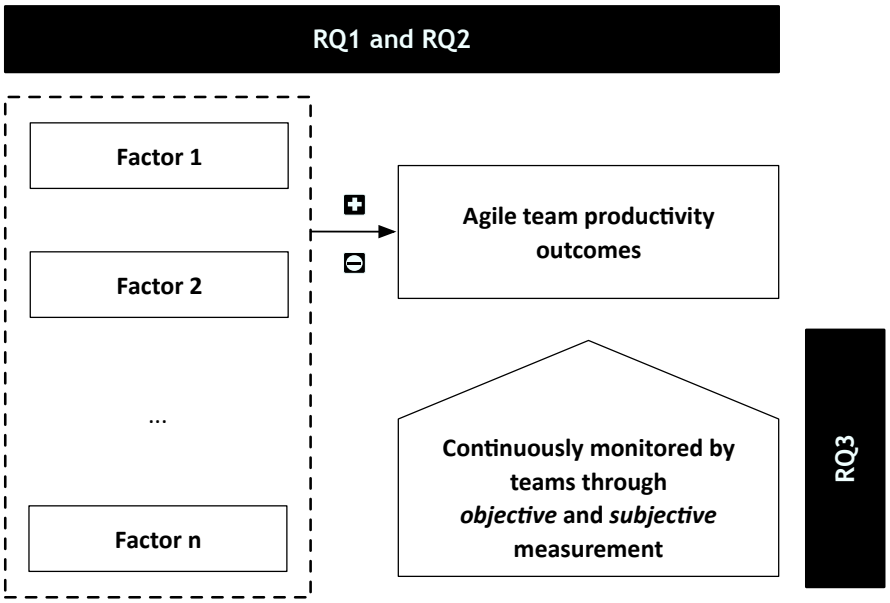

Figure 1.1: *Research questions and our high-level research framework*

Company A is a large Brazilian financial corporation with over 500 IT employees, which had previously used plan-driven development processes. The company managers decided to adopt agile methods to increase team productivity, and they had been using them for two years. The organizational structure and coordination are primarily vertical, a concept described by Galbraith [1973], where project managers usually implement coordination processes. Project A is a re-development of an existing system for the financial market involving several institutions. The project started in March 2010 and was estimated to last for approximately two years. The team adopted several XP and Scrum practices and used one-week iterations.

Company B has been delivering e-commerce and infrastructure services for over ten years and has used only agile methods to develop software. It employs approximately 120 developers. The organizational structure and coordination are primarily horizontal [Galbraith, 1973], where coordination processes are usually provided by an individual team member who communicates directly with other members or users on a one-to-one basis. Project B is a new development of an e-commerce service in a market with other competitors. The project also started in March 2010, but does not have a specific deadline, as they are developing software as a service, with continuous improvement and new functionalities. The project adopts several XP, Scrum, and Lean principles and practices.

Company C is an important player in Internet content and access provision in Brazil. The organizational structure and coordination are primarily vertical [Galbraith, 1973], but the hierarchy is smaller than Company A. The IT department employs approximately 200 developers and had also previously used plan-driven development processes. They have applied agile methods since 2008. Project C is the maintenance of a recommendation system for products from several virtual stores. The project adopts mainly Scrum practices and some XP and Lean principles and practices.

Company D is a multinational company working with cutting-edge software technologies and processes, with a local presence in Brazil. The organizational structure and coordination are primarily horizontal [Galbraith, 1973], and it is flatter than Company B. Project D is a web-based solution for digital content, built upon a content management system. The company is one of the most influential agile methods players in the market, disseminating a culture based

on sustainable business, technical excellence and social justice. However, as a consultancy, the projects are not only immersed in its own culture, but also in customers' culture, that may differ from project to project.

Regarding the researcher experience, I had over five years of experience working with agile practices and twelve years of experience in software development prior to this Ph.D. research. I have worked in Company A, where I helped to deploy agile methods between 2008 and 2009. Thus, the data collection there was easier due to the previous relationship. I had no previous experience working in Companies B and C, but we established a good research collaboration, particularly in Company B. Finally, I joined Company D during the Ph.D. as a project manager, which made the data collection easier in the Research Phase III.

We also performed a national survey involving many companies over the country, varying in size, domain, region etc. Our non-deterministic sample better covers the Southeast and Midwest regions. However, we consider the sample representative of the other Brazilian regions, since its distribution was not far below the real distribution (see Chapter 3, Section 5.1). Therefore, this research is being carried out under an intensive partnership with Brazilian IT industry.

Finally, we collaborated with other universities in the scope of this research, mainly funded by FAPESP. Part of the research was also funded by the Research Council of Norway (grant 202657), which started and terminated in 2011. The goal was to conduct multiple case studies in Brazil and analyse the data at Norwegian University of Science and Technology - NTNU[2]. Prof. Reidar Conradi's research group has large experience on empirical studies and agile methods[3][4].

Some of these studies were performed in collaboration with organizations. We thus performed an empirical study inside companies A, B, and C, within a real environment and with real problems. We collected qualitative and quantitative data before the research visit to Norway. We analyzed and reported results of this phase in collaboration with Dr. Daniela Cruzes and Professor Reidar Conradi (see Section 1.5). Professor Tore Dybå also provided valuable feedback during the visit. This research project is also funded by CNPq (grant 76661/2010-2), in collaboration with CIn-UFPE's process management and improvement group[5].

1.4 Research design

Empirical studies may apply different research methods. This research is a result of the combination of qualitative and quantitative methods, specifically case studies, survey, and action research. These three research methods are appropriate to answer research questions focused on "what" and "how" [Yin, 2008].

The dominant philosophical position taken in this thesis is *interpretivism* [Klein and Myers, 1999], which recognizes that scientific knowledge cannot be separated from its human context. We also adopt *pragmatism* [Menand, 1997], in which knowledge is judged by how useful it is for solving practical problems. The research questions guide the research design towards *exploratory* and *explanatory* inquiry strategies. Figure 1.2 summarizes the research phases, the methods used and their purposes.

Phase I is a combination of literature review of software productivity and warm-up studies. I was working already with data from Company A and performed a preliminary analysis of the effects of agile methods on their software productivity. However, traditional software productivity analysis, such as recommended by ISO/IEC 15393[6] can generate misleading analyses [Kitchenham *et al.*, 2007]. There were not similar agile projects to compare with at that time, just past projects developed in plan-driven approaches. Productivity definitions were based on traditional software

[2] www.ntnu.no
[3] http://www.idi.ntnu.no/~conradi
[4] http://www.sintef.no/Kontakt-oss/Alle-ansatte/?EmpId=302
[5] http://www.cin.ufpe.br/~promise/
[6] ISO/IEC 15939: Software Engineering—Software Measurement Process, Int'l Organization for Standardization, 2001

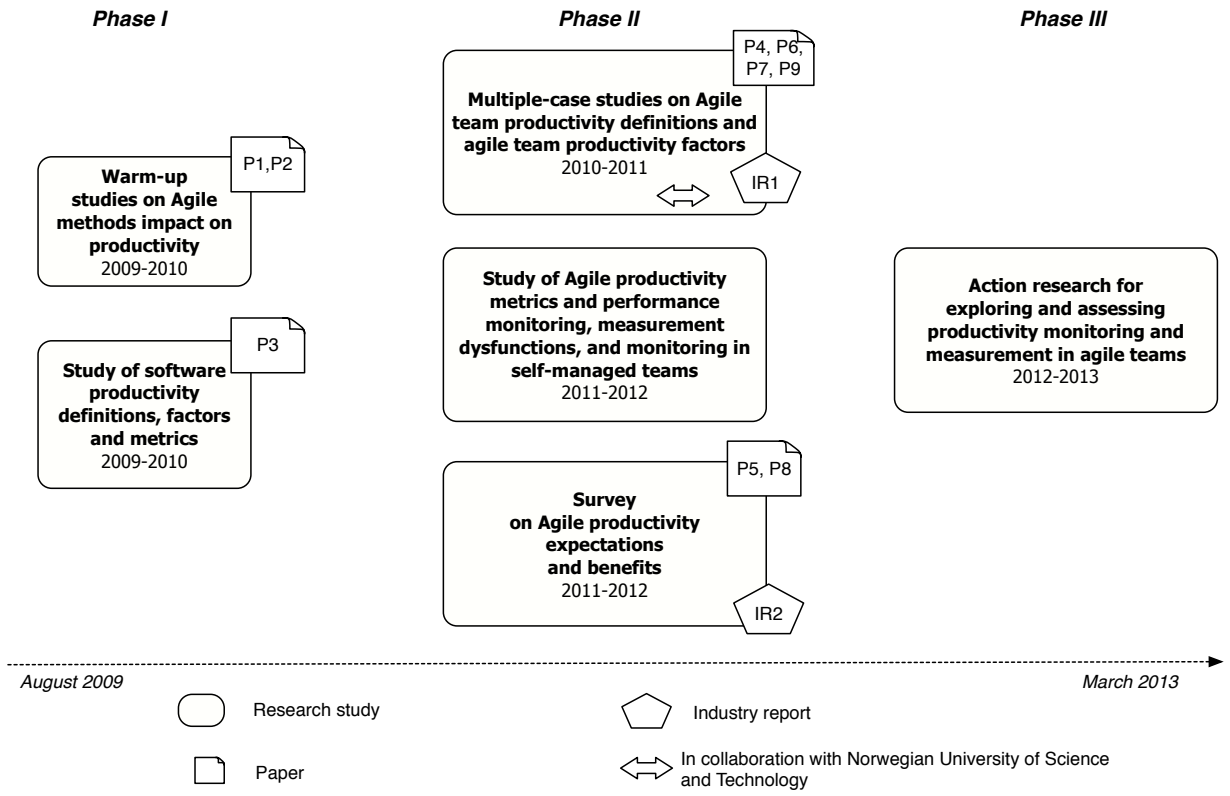

Figure 1.2: *Research phases*

development approaches, raising questions on the validity of the metrics adopted by the company to evaluate agile projects. In parallel, we conducted a literature review based on primary and secondary studies on software productivity definitions, factors, and metrics, both in traditional and agile approaches.

Phase II comprises two types of studies: case studies and surveys. We designed multiple case studies to explore productivity definitions and factors in agile teams. The survey was developed to generate evidence on the importance of productivity for agile teams. After that, we collected qualitative data on how some companies are evaluating agile teams productivity, and if they are using metrics for that. Results were integrated in three aspects: agile productivity definitions, productivity importance and expectations in agile teams, and factors impacting agile team productivity. Our findings answer RQ1 and RQ2.

Finally, Phase III involves carrying out an agile team productivity monitoring process, and the evaluation of its usefulness. Since agile companies (and teams) are still trying to find good ways to monitor their productivity, we designed an action research study. The main goal is not to develop new metrics for measuring productivity of agile teams, but to iteratively adopt some proposed agile metrics and evaluate them in a real setting. Our findings answer RQ3.

1.5 Papers, Technical/Industry reports, and Book Chapters

We generated a number of research papers (P), technical/industry reports (IR), and book chapters (CH) based on this research.

P1 MELO, C. O.; FERREIRA, G. R. M. Adopting Agile in a Large Government Institution: a case study (in Portuguese). In: Workshop Brasileiro de Métodos Ágeis (WBMA), Conferência Brasileira sobre Métodos Ágeis de Desenvolvimento de Software (Agile Brazil 2010). Porto Alegre. p. 104-117.

P2 MELO, C. O.; SANTOS Jr., C. D. ; FERREIRA, G. R. M. ; KON, F. An exploratory study of factors associated with learning in agile teams on industry (in Portuguese). In: VII Experimental Software Engineering Latin American Workshop, 2010, Goiânia.

P3 MELO, C. O.; KON, F. Empirical evaluation of agile practices impact on team productivity. In: 12th International Conference on Agile Software Development (XP), Doctoral Symposium, Madrid, 2011, pp. 322-323.

P4 MELO, C. O.; CRUZES, D. S. ; KON, F. ; CONRADI, R. Agile Team Perceptions of Productivity Factors. In: Proceedings of the Agile Development Conference (AGILE), Salt Lake City, USA, 2011, pp. 57-66.

P5 CORBUCCI, H.; GOLDMAN, A. ; KATAYAMA, E. ; KON, F. ; MELO, C. O. ; SANTOS, V. S.. Genesis and Evolution of the Agile Movement in Brazil: a perspective from the Academia and the Industry. In: Proceedings of 25th Brazilian Symposium on Software Engineering (SBES), 2011, pp. 98-107.

P6 MELO, C. O.; KON, F. Productivity of agile teams (in Portuguese). Software Engineering Magazine, Brazil, v. 43, p. 1 - 9, 05 dez. 2011.

P7 MELO, C. O.; CRUZES, D. S. ; KON, F. ; CONRADI, R. Interpretative Case Studies on Agile Team Productivity and Management. *Information & Software Technology* 55(2), 2013, pp. 412-427.

P8 MELO, C. O.; KATAYAMA, E.; SILVA, V. S.; CORBUCCI, H.; PRIKLADNICKI, R. GOLDMAN, A.;KON, F. The evolution of agile software development in Brazil. *Journal of Brazilian Computer Society* 19(4), 2013, pp. 523-552.

P9 MELO, C. O.; SANTANA, C.; KON, F. Developers motivation in agile teams. 38th Euromicro Conference on Software Engineering and Advanced Applications (SEAA), Çesme, Izmir, 2012, p. 376-383.

IR1 MELO, C. O.; KON, F. Productivity Factors in Agile teams - an exploratory study in Brazilian Companies (in Portuguese). Industry report. March, 2012. Available at: http://agilcoop.org.br/ProdutividadeTimesAgeis.

IR2 MELO, C. O.; SANTOS, V. A.; CORBUCCI, H.; KATAYAMA, E.; GOLDMAN, A.; KON, F. Agile methods in Brazil: state of the practice in teams and organizations (in Portuguese). Technical Report MAC-2012-03. Department of Computer Science. IME-USP. May, 2012. Available at: http://agilcoop.org.br/MetodosAgeisBrasil2011, *with over 6736 visits* (April, 2015).

CH1 GOLDMAN, A; MELO, C. O. ; KON, F.; CORBUCCI, H.; SANTOS, V. A História das Metodologias Ágeis no Brasil, In: *Métodos Ágeis Para Desenvolvimento De Software*. Bookman, 1a edição, v. 1, p. 16-20, 2014.

CH2 MELO, C. O. Productividade e adaptabilidade de times ágeis: conceitos e paradoxos, In: *ThoughtWorks Antologia Brasil: Histórias de aprendizado e inovação.*.Casa do Código, 1a edição, v.1. p. 1-10, 2014.

In addition, three articles and two book chapters were published in collaboration with other reseachers, covering diverse themes related to the topics of this thesis.

P10 SOUSA, T. C.; MELO, C. O. Geração de Testes de Aceitação em Fit a partir de Especifações em B. In: IV Workshop de Desenvolvimento Rápido de Aplicações do Simpósio Brasileiro de Qualidade de Software (WDRA-SBQS), 2010, p. 1-8.

P11 TAKEMURA, C.; MELO, C. O. Studying agile organizational design to sustain innovation. In: Agile Brazil, 2012, São Paulo. Proceedings of the III Brazilian Workshop on Agile Methods (WBMA 2012), 2012. p. 13-24.

P12 DE MELO OLIVEIRA, RENAN ; GOLDMAN, ALFREDO ; MELO, C. O. Designing and Managing Agile Informative Workspaces: Discovering and Exploring Patterns. In: 2013 46th Hawaii International Conference on System Sciences (HICSS), p. 4790-4799, Wailea, 2013.

CH3 Bertholdo, Ana Paula O. ; da Silva, Tiago Silva ; MELO, C. O.; KON, FABIO ; Silveira, Milene Selbach. Agile Usability Patterns for UCD Early Stages. Lecture Notes in Computer Science. Ied.: Springer International Publishing, 2014, v. 8517, p. 33-44.

CH4 Silva, Tiago Silva; Silveira, Milene Selbach; MELO, C. O.; Parzianello, Luiz Claudio. Understanding the UX Designer's Role within Agile Teams. Lecture Notes in Computer Science. Ied.: Springer Berlin Heidelberg, 2013, v. 8012, p. 599-609.

1.6 Claimed Contributions

The goal of this thesis is to investigate productivity concepts, importance, factors, and monitoring approaches in agile teams. Based on this objective, we claim that this thesis contains five main novel contributions which are:

C1. Empirical verification of the importance of productivity for companies adopting agile and the perceived benefits Agile methods are being widely adopted by companies worldwide. The main claimed reasons for this adoption are: to accelerate time to market, to enhance the ability to manage changing priorities and to increase productivity. Our empirical results verify this assumption in the Brazilian IT industry, in which 481 practitioners responded to our survey. Companies not only adopt agile expecting productivity improvement, but also perceive productivity as one of the most important benefits. There is no previous literature on productivity expectations in Brazilian agile teams. Productivity is, therefore, an important issue when deploying, analyzing, and improving agile methods in companies. Despite the result is limited to the Brazilian scenario, we argue that our country study enable comparisons among countries, enhancing the knowledge on productivity in a particular way.

C2. Rationale on productivity definition in the agile methods context. There are several concepts involved in the productivity definition, such as effectiveness, efficiency, and performance. To date, there is no discussion around this definition in agile software development teams. In fact, we found that productivity is often a diffuse term for agile team members. One may argue that productivity is not the focus of agile teams. Indeed, there is a paradoxical relationship between flexibility and efficiency in the context of agility and productivity. We thus provide a rational in which both flexibility and efficiency are important for long-term productivity in agile teams. We also argue that agile team productivity should be evaluated in a holistic way. Software development is knowledge work, which involves many other productivity dimensions, such as creativity, innovativeness, and learning.

C3. Empirical verification of agile team productivity factors. Software development productivity is influenced by many different factors (so-called productivity factors). We have conducted a multiple-industrial case study of agile teams in three large Brazilian companies for six months. The results presented here are based on detailed and rigorous investigations of three agile teams. The findings shed light on under-researched questions pertaining to the productivity factors of agile teams. Our contribution to research in this area has been to synthesize an I-P-O (input-process-output) conceptual framework on factors impacting agile team productivity and to present a thematic map exploring productivity factors impacting agile teams, whereas the main factor was *team management*. Our study sheds light on staff

Table 1.1: *Main contributions, research questions, and publications*

Contribution	Research question	Publications
C1. Empirical verification of the importance of productivity for companies adopting agile, and perceived benefits.	RQ1	P5, P8, IR2
C2. Rationale on productivity definition in agile methods context.	RQ1	P3, P4, P6, P7
C3. Empirical verification of agile team productivity factors.	RQ2	P4, P7, IR1
C3.1. Motivators in agile teams.	RQ2	P9
C4. A framework of agile team productivity factors and their impact, to be tested.	RQ2	P7
C5. A case on team productivity monitoring process considering adaptability and evaluation of agile team productivity metrics' usefulness.	RQ3	CH2

turnover, team design choices, and inter-team coordination factors that, through processes mediators, generated variations on the observed productivity outcomes.

C3.1. As a minor contribution, we analyzed motivational factors in agile teams impacting on productivity [Melo *et al.*, 2012]. We contrasted our results with a recent theoretical framework on software engineer's motivation proposed by Sharp *et al.* [2009].

C4. A framework of agile team productivity factors and their impact, to be tested. To interpret data in the case studies, we adapted three IPO frameworks of teamwork effectiveness respectively from Cohen and Bailey [1997], Yeatts and Hyten [1998], and Marks *et al.* [2001], aiming to describe a more coherent conceptual framework of agile team productivity. Based on these proposed frameworks, we selected inputs, processes, and outcomes related to the agile values and principles [Beck *et al.*, 2001]. After the study (corresponding to papers P4 and P7), we revised the conceptual IPO framework by referring back to the findings on team management. The updated framework (with agile productivity factors, mediators, and their impact) is prepared for testing through confirmatory studies.

C5. A team productivity monitoring process considering agility and an evaluation of the usefulness of agile team productivity metrics. We develop, in collaboration with Company D a team productivity monitoring process for agile teams. The team discusses their goals and defines main productivity areas they want to monitor. We assess the environment and the team, supporting the choice of metrics. The team implements the metrics in the work environment, while we observe their use. In the retrospectives and project meetings, we discuss the metrics usefulness, the actual context and goals, and decide if the team will keep the same metrics or not. The main contribution here is to provide an agile productivity monitoring model that can be further tested by other agile teams. Finally, we aim at understanding how agile productivity metrics support productivity monitoring, and whether they provide a tool for diagnostic or not.

Table 1.1 summarizes the preliminary contributions, and their relationship with the posed research questions and papers.

1.7 Thesis structure

This proposal consists of two parts:

Part 1 contains three chapters and aims to provide an introduction to the field, place the research into context, and present the overall research design. Chapters 2 and 3 are an introduction to the field of agile methods and software productivity, respectively. Chapter 2 describes the theoretical principles of agile methods, most frequent claimed benefits, the concept of agility, and some paradoxes in the agile software development definition. Chapter 3 presents an overview of software productivity definitions and a tentative conciliation considering the agile methods context. It also describes productivity factors, monitoring and measurement, both in traditional and agile paradigms. Chapter 4 discusses research approaches and strategies, as well as advantages and challenges in the approaches adopted in this thesis: case studies, survey, and action research.

Part 2 contains four chapters and aims at providing detailed results and discussions of the studies. Chapter 5 presents the research design for RQ1, our results, and discusses implications for theory and practice. Chapter 6 depicts the research design for RQ1 and RQ2, discusses our main findings in light of the existing literature, as well as implications for research and practice, and limitations. Chapter 7 describes the research design for RQ3, our results, discussion, and limitations. The thesis is summarized in Chapter 8 and future work is proposed.

Chapter 2

Agile software development

Nowadays, software development is more technically complex; more strategic; and brings to the fore the divergent viewpoints of a wider variety of stakeholders [Nerur *et al.*, 2010]. Software development is a complex activity characterized by tasks and requirements that exhibit a high degree of variability [Boehm and Turner, 2005, Nerur *et al.*, 2005]. This leads to the inescapable conclusion that software development, both today and in the future, is and will continue to be undertaken in a much more uncertain context [Nerur *et al.*, 2010].

Agile methods emerged to address many of these issues and have been increasingly adopted and "rapidly joined the mainstream of development approaches" [West and Grant, 2010]. In the foreword to the "Agile Software Development" book [Dingsøyr *et al.*, 2010], H. Erdogmus states that "agile software development is the most important paradigm that has swept the software development world over the last decade. Even if it does not represent the most popular software development approach in actual use, it has certainly become one of the most talked about. Its vocabulary and prime ideas have already started spilling over into other fields, project management in particular". He also recognizes that agile is a multi-faceted and poorly delimited concept, dependent on personal and contextual interpretations. This certainly makes agile methods a hard topic to conduct research on, because researchers should try to avoid misconceptions.

Agile methods promise to achieve high productivity and deliver high-quality software, attracting the attention of companies, which demand ever-higher development speed and quality in their products. Typically, the following benefits are advocated [Highsmith, 2007, Kettunen, 2009, Schwaber, 2007]:

- increased customer satisfaction;
- reduced time-to-market (better "time-to-benefit");
- increased quality;
- improved project portfolio and product management (project types, features);
- improved product development investment management (control and flexibility);
- reduced "waste" (increased efficiency, productivity, development cost);
- better predictability (visibility);
- better risk management (risk reduction);
- better workforce morale (developer satisfaction, well-being).

Despite the growing body of empirical evidence on those advantages, it is still insufficient to be fully conclusive [Dybå and Dingsoyr, 2008, Kettunen, 2009]. In spite of that, the claimed benefits of agile development match well with the typical problems of turbulent software development environments, such as time-to-market pressures, productivity demands, fluid and ambiguous product requirements, and changing environment [Baskerville *et al.*, 2006].

In this chapter, we give a short overview of the agile software development paradigm, focusing on the main theoretical changes in comparison to traditional development methods. We also summarize the *agility* construct and its relationship with software productivity. Finally, we here use *agile method* and *agile development* as synonyms.

2.1 The Agile software development paradigm

The birth of the agile movement around the year 2000 has strong roots in the history of software engineering. The agile ideas echoed previous works such as *The Mythical Man-Month: Essays on Software Engineering* by Brooks [1975] and the concept of Rapid Prototyping [Naumann and Jenkins, 1982]. They were a consequence of a variety of factors, ideas, and proposed best practices that arose mainly in the context of the object-oriented programming community. Even being a remarkable change in software development thinking, these ideas have been around since the 1970s or even before as explained by Abbas *et al.* [2008]. Because they were not treated seriously enough, it took about 30 years for them to be recognized as an effective way to develop software [Corbucci *et al.*, 2011].

During the 1990s, a combination of factors produced a fertile land for the growth of agile ideas [Abbas *et al.*, 2008]. First, there was a reaction to heavyweight, prescriptive approaches for developing software. Second, the increasing level of change in the business environment urged practitioners to handle complex and unpredictable requirements in systems development. Thus, practitioners started to revisit old alternative ways of developing software such as iterative and incremental development and close customer involvement proposed by Winston Royce [1970], test-first development and continuous integration used by NASA's 1961-63 Project Mercury described by Larman and Basili [2003], and rapid prototyping defined by Naumann and Jenkins [1982].

In the second half of the 1990s, research and practical results in the fields of object-oriented programming, design patterns, automated testing, refactoring, and the like, produced a common mind set that drove the definition of multiple software development methods that had the core agile principles in common. These methods include Extreme Programming (XP), Scrum, Dynamic Systems Development Method (DSDM), Adaptive Software Development, Crystal, Feature-Driven Development, Pragmatic Programming, and others.

In early 2001, a group of independent practitioners with a strong link with the software industry and a weaker, but still relevant, link with research groups from academia decided to join forces and founded what was later called the agile movement. To make these ideas more concrete, 17 software experts met from February 11th to 13th in the mountains of Utah, USA, to collectively craft the agile manifesto [Beck *et al.*, 2001]. The goal of the manifesto was to bring attention to the idea that, to produce high-quality, valuable software, development teams must focus on (1) individuals and interactions, (2) working software, (3) customer collaboration, and (4) responding to change. Those points were presented as more important than emphasizing processes and tools, comprehensive documentation, contract negotiation, and following previously defined plans. Agile methods, therefore, are a reaction to more traditional methods (known as plan-driven) [Nerur *et al.*, 2005], also known as a change-driven development paradigm [Moe, 2011].

According to Highsmith [2002], agile development is not defined by a small set of practices and techniques. Agile development defines a strategic capability, a capability to create and respond to change, a capability to balance flexibility and structure, a capability to draw creativity and innovation out of a development team, and a capability to lead organizations through turbulence and uncertainty.

Nerur *et al.* [2010], however, make a counterpoint on the underlying assumptions of agile methods novelty. Concepts such as iterative development, learning, self-organization, reflective practice, self-directed teams and stakeholder participation, have evolved separately in other disciplines, and one or more of them were used in some form in software development as well. They claimed that this recently "emerged" paradigm can be interpreted as a re-combination of earlier viewpoints, and it is not altogether surprising. Nevertheless, in our view, this re-combination *per se* represents a major

shift in how practitioners develop and manage software development teams and projects.

2.2 Theoretical differences between agile and other paradigms

Adopting an agile approach entails changing aspects of the organization, including its structure, culture, and management practices, work procedures, tools and techniques, communication channels, problem-solving strategies, and roles of people [Moe, 2011, Nerur *et al.*, 2005].

A rationalized, engineering-based approach has dominated software development almost since its inception [Nerur *et al.*, 2005]. Extensive upfront planning is the basis for predicting, measuring, and controlling problems and variations during the development life cycle. The traditional software development approach is process-centric, guided by the belief that sources of variations are identifiable and may be eliminated by continually measuring and refining processes [Highsmith and Cockburn, 2001]. The primary focus is on conducting optimized and repeatable processes, in which planning and control are accomplished by a command and control style of management.

In the traditional approach, software development is usually guided by life cycle models such as the waterfall, spiral, or some variations of these [Nerur *et al.*, 2005]. These models specify the tasks and related roles to be performed and the desired outcomes of each phase. In addition to the end product of working code, these methods also produce a large amount of documentation that codifies process and product knowledge. Communication among project participants is also formalized through these documents (although it also happens in other ways). Customers play an important role during specification and validation, but their participation is minimal in other activities [Nerur *et al.*, 2005]. This traditional school fosters single-loop learning [Argyris and Schön, 1978], or "adaptive learning" [Senge, 1990], in which problem solving mechanisms determine goal-seeking behavior [Nerur *et al.*, 2005].

On the other hand, agile development deals with unpredictability by relying on people and their creativity rather than on processes [Cockburn and Highsmith, 2001]. It is characterized by short iterative cycles of development driven by prioritized features, periods of reflection and introspection, collaborative decision-making, incorporation of rapid feedback and change, and continuous integration of code changes into the software under development. The team works in small pieces of the product, each of which involving planning, development, integration, testing, and delivery. Developers work with customers as active team members, deciding together the highest priority features, which enables a collaborative decision-making.

Agile methods encourage a leadership-and-collaboration style of management where the project manager's role is that of a facilitator or coordinator [Nerur *et al.*, 2005]. Each development cycle aims to deliver useful working code to the customer. Agile methods discourage extensive documentation beyond the code [Nerur *et al.*, 2005]. Product knowledge, therefore, becomes tacit. Rotation of team membership ensures this knowledge is not restricted to a few individuals. Agile development is characterized by social inquiry in which extensive collaboration and communication provide the basis for collective action [Nerur *et al.*, 2005]. Diverse stakeholders including developers and end users go through repeated cycles of thought-action-reflection that foster an environment of learning and adaptation. This constant re-evaluation of discrepancies against existing norms and values fosters what is called "double-loop learning", which is "generative" by increasing both learning and the ability to innovate and use change to one's advantage [Senge, 1990]. Team members, empowered with more discretionary and decision-making powers, are not confined to a specialized role, enabling also self-organization and responsiveness in new surprising situations [Nerur *et al.*, 2005].

Usually, traditional software process improvement (SPI) approaches support the ideology of creating and improving a universal and repeatable software development process for an organization. SPI initiatives are largely controlled by the organizational level and management [Lycett *et al.*, 2003, Salo and Abrahamsson, 2007]. In contrast, the principles of agile software development require that "at regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly" [Beck *et al.*, 2001]. The main traditional improvement practices are focused on supporting the improvement at the organizational level, aiming to conduct better software projects.

On the other hand, in agile development, the primary focus of process adaptation within agile project teams is on the immediate use of the experiences of developers in improving the ongoing project iteratively [Salo and Abrahamsson, 2007].

Table 2.1 summarizes the main theoretical differences between the traditional and agile approaches adapted from Nerur *et al.* [2005], Nerur and Balijepally [2007], and Salo and Abrahamsson [2007].

Table 2.1: *Theoretical differences between Traditional and Agile approaches*

	Traditional View of Design	**Emergent Metaphor of Design**
Fundamental assumptions	Systems are fully specifiable, predictable, and can be built through meticulous and extensive planning	Small teams using the principles of continuous design improvement can develop high-quality, adaptive software and testing based on rapid feedback and change
Control	Process centric	People centric
Knowledge Management	Explicit	Tacit
Role Assignment	Individual: favors specialization	Self-organizing teams: encourages role interchangeability
Customer's Role	Important	Critical
Desired Organizational Form/Structure	Mechanistic (bureaucratic with high formalization)	Organic (flexible and participative encouraging cooperative social action)
Design process	Deliberate and formal, linear sequence of steps, separate formulation and implementation, rule-driven	Emergent, iterative and exploratory, knowing and action inseparable, beyond formal rules
Goal	Optimization	Adaptation, flexibility, responsiveness
Problem-solving approach	Selection of best means to accomplish a given end through well-planned, formalized activities	Learning through experimentation and introspection, constantly reframing the problem and its solution
View of the environment	Stable, predictable	Turbulent, difficult to predict
Type of learning	Single-loop/adaptive	Double-loop/generative
Key characteristics	Control and direction Avoids conflict Formalizes innovation Manager is controller Design precedes implementation	Collaboration and communication Embraces conflict and dialectics Encourages exploration and creativity and is opportunistic Manager is facilitator Design and implementation are inseparable and evolve iteratively
Immediate focus of process improvement	Improvement of organizational software development processes/(future projects)	Iterative Improvement of daily working practices of ongoing project
Theoretical and/or philosophical roots	Logical positivism, scientific method	Action learning theory, Dewey's pragmatism, phenomenology

2.3 The agility concept

The concept of agility within the context of business was first introduced in a study conducted by the Iaccoca Institute [Goldman *et al.*, 1991]. In the manufacturing field, agility includes being

able to bring new products to market and respond to customer needs rapidly by incorporating innovative management structures, flexible technology, and a skill base of knowledgeable workers [Bonner *et al.*, 2010, Kettunen, 2009].

Prior literature provides various definitions and descriptions of software development agility. While the literature is vague about the underlying dimensions and measures of software development agility, there is a common theme underlying the various definitions and descriptions in that agility is generally defined in terms of embracing and responding to change [Lee and Xia, 2010]. Table 2.2 describes the main definitions of agility related to agile software development, adapted from Conboy [2009] and Lee and Xia [2010].

Table 2.2: *Software development agility*

Literature	Relevant Definitions/Concepts/Ideas
Conboy and Fitzgerald [2004]	Agility is defined as the continual readiness of an entity to rapidly or inherently, proactively or reactively, embrace change, through high-quality, simplistic, economical components and relationships with its environment
Highsmith [2004]	Agility is the ability to both create and respond to change in order to profit in a turbulent business environment; it is the ability to balance flexibility and stability.
Larman and Basili [2003]	Agility is rapid and flexible response to change.
Erickson *et al.* [2005]	Agility is associated with such related concepts as nimbleness, suppleness, quickness, dexterity, liveliness, or alertness; it means to strip away the heaviness in traditional software development methodologies to promote quick response to changing environments and changes in user requirements.
Henderson-Sellers and Serour [2005]	Agility refers to readiness for action or change; it has two dimensions: (1) the ability to adapt to various changes and (2) the ability to fine-tune and reengineer software development processes when needed.
Lyytinen and Rose [2006]	Agility is defined as the ability to sense and respond swiftly to technical changes and new business opportunities; it is enacted by exploration-based learning and exploitation-based learning.
Cockburn [2006]	Agility is being light, barely sufficient, and maneuverable.
Qumer and Henderson-Sellers [2008]	Agility is a persistent behavior or ability of an entity that exhibits flexibility to accommodate expected or unexpected changes rapidly, follows the shortest time span, and uses economical, simple, and quality instruments in a dynamic environment; agility can be evaluated by flexibility, speed, leanness, learning, and responsiveness.
Kettunen [2009]	Agility strives to increase the quality and service factors by meeting the current customer requirements by being flexible to customer demands and market changes.
Conboy [2009]	Agility is defined as the continual readiness of an information system development method to rapidly or inherently create change, proactively or reactively embrace change, and learn from change while contributing to perceived customer value (economy, quality, and simplicity), through its collective components and relationships with its environment.

As summarized in Table 2.2, flexibility can be interpreted as one enabler for agility. But flexibility alone may not be enough for comprehensive agility [Kettunen, 2009]. Agile development promotes both response extensiveness in terms of embracing different types of changes [Henderson-Sellers and Serour, 2005, Qumer and Henderson-Sellers, 2008] and response efficiency in terms of high speed

and low cost [Conboy and Fitzgerald, 2004, Erickson *et al.*, 2005, Larman and Basili, 2003, Qumer and Henderson-Sellers, 2008]. There is also dimensions such as exploration and exploitation-based learning, and learning from change [Conboy, 2009, Lyytinen and Rose, 2006, Qumer and Henderson-Sellers, 2008].

Wang and Conboy [2009] also studied how agility is manifested in agile software development taking the key concepts of CAS (Complex Adaptive Systems) as theoretical lenses. They investigated the meaning of agility from three facets: autonomous but sharing team, stability with embraced uncertainty, and team learning. They found that stability and discipline are as desirable as flexibility, and context sharing is of the same value and importance as knowledge sharing in agile processes. In addition, the collective nature of learning is underlined.

Considering these many agility definitions, it is important to establish a minimal conceptual construct that enables us to interpret agility in other software development dimensions, such as productivity.

2.3.1 Attributes of agility

Qumer and Henderson-Sellers [2006] suggest that an agile software development method may be described by the attributes of agility: flexibility, speed, leanness, responsiveness, and learning. We adopt their taxonomy to establish a conceptualization of agility that will be useful in our research. We describe the taxonomy below based on Qumer and Henderson-Sellers [2006] and Qumer and Henderson-Sellers [2008], adding also explanations from Conboy [2009] .

Flexibility is the ability or behaviour (flexible) of an entity that enables adapting to changes whenever it is required. A method or phase in a method may demonstrate flexibility by accommodating expected or unexpected changes. It is "the ability of a software development method to create change, or proactively, reactively, or inherently embrace change in a timely manner, through its internal components and relationships with its environment" [Conboy, 2009]. It is often related to the adaptability capability.

Speed of an entity characterizes rapid and quick behaviour to get to the desired destination or to achieve goals. A speedy method may help to show the results quickly by following a specific approach.

Leanness refers to compactness and tidiness. A lean method gives the desired quality output, economically, in the shortest possible time frame by applying simple and quality means of development. It is "the maximization of value through economy, quality, and simplicity" [Conboy, 2009].

Learning refers to knowledge and improvement and is an indispensable ability of an entity, which is achieved primarily by using up-to-date knowledge and experience, gained from previous practices. A learning method shows continuous improvement over time.

Responsiveness refers to life, reaction, and sensitivity. A responsive method is one that does not remain silent when response is required in different situations. It refers to the fact that not only does it become easy to accept the changes but the changes must be reflected and visible.

Conboy [2009] states that agility taxonomies should be extended by developing a set of metrics to evaluate actual performance outcomes under each component of the taxonomies (he has also proposed one). It is well known that behaviors are not always rewarded by positive outcomes, so applying outcome measures of software development agility across methods, method variants, organizations, and projects could reveal some very interesting insights and add credibility to those who claim their methods or practices are agile.

With this predefined dimensions of agility, we aim to interpret the impact of the agile methods paradigm on software productivity evaluation in general, and team productivity in particular. We do not plan to evaluate the agility degree of companies participating in the research. We seek to

explore agility properties to better understand how to define and evaluate software productivity in such highly flexible, adaptive, and responsive environment. We also aim to check if there is any common dimension between software development agility and software development productivity. We explore better this comparison in Chapter 3, but we draw our first insights into the question in this chapter.

It is important to notice, however, that there is already a need for a better understanding of what constitutes agility [Abrahamsson *et al.*, 2009]. Proposed agility taxonomies (e.g., [Conboy, 2009, Qumer and Henderson-Sellers, 2006] are already in study by researchers. Therefore, we are working with a concept that is somehow not well delimited.

2.4 Paradoxical perspectives on Agile Software Development and Software productivity

A paradoxical view – distinct from a dichotomy or even a dualist way of dealing with contradictions – emphasizes the coexistence of the competing themes in a contradiction and makes use of the tensions the contradictions generate rather than trying to eliminate them [Wang and Conboy, 2009]

When taking a paradoxical view of agile methods, we can identify many competing definitions. For instance, agile methods contain planning activities. However, agile champions, such as Highsmith, view planning as a paradox in an adaptive environment, since outcomes are naturally unpredictable [Highsmith, 1997]. Wang and Conboy [2009] also argue that an agile process is a planning-driven process geared to responding to change, which is also a paradoxical perspective.

Project visibility is also a paradox, according to Lindstrom and Jeffries [2004]. On the one hand, with so much visibility, customers are in a position to cancel the project if progress is not sufficient. On the other hand, progress is so visible, and the ability to decide what will be done next is so complete, that XP projects tend to deliver more than what is needed, with less pressure and stress. Other paradoxes were identified on agile principles and practices, such as group cohesion [McAvoy and Butler, 2007], Test Driven Development (TDD) [George and Williams, 2004], Architecture [Kruchten, 2010], agility through discipline [Beck and Boehm, 2003], and requirements engineering [Schwaber, 2002].

In the software productivity literature, there is little research on paradoxes when adopting agile methods. We can find studies recognizing the productivity paradox (also known as productivity dilemma) in the manufacturing industry [Adler *et al.*, 2009, Helo, 2004, Peng *et al.*, 2008]. Recently, Lee and Xia [2010] recognized that the agile literature tends to take a simplistic view on the tension between software development agility and development performance. It views agility to be universally desirable and does not recognize differential effects of agility on different aspects of development performance. Lee and Xia [2010] found that software teams should take care when implementing agile principles such as "welcome changing requirements even late in development", because when time and cost are top priorities, teams can be better off by selectively responding to changing requirements and thus increasing response efficiency. Therefore, some dimensions of productivity (better discussed in Chapter 3), such as efficiency, cannot always be neglected in favor of agility properties, such as flexibility.

2.4.1 Agility and the productivity paradox

It has been argued that flexible development processes are associated with better-performing projects in a highly uncertain and dynamic environment [MacCormack *et al.*, 2001]. In this sense, one may argue that productivity is not the focus of agile teams, but the flexibility. Studies in the agile development field (e.g., [Iivari and Iivari, 2011]) argue that agile development orientation avoids the rational culture, which focus on productivity, efficiency, and goal achievement. In this discourse, agile development would be oriented toward a developmental culture, which is characterized by flexibility and external focus. In this way, agile companies and teams would perform better by

adopting flexibility and external focus as top priorities. But how does productivity, an important and desirable outcome of teams and organizations, is positioned in this context?

Abernathy [1978] highlighted the inconsistencies between activities focused on productivity improvements and cost reductions and those focused on innovation and flexibility. He questioned whether it is possible for organizations to pursue both types of activities simultaneously. Abernathy's productivity dilemma arises because mature processes provide few natural opportunities for learning. In mature processes, activities proceed according to a plan, exceptions are rare, and external stimuli are filtered and buffered to prevent them from disrupting routine operation. When processes are stable and predictable, organizational activity validates existing knowledge, but provides no new information to enable knowledge creation [Adler *et al.*, 2009, Peng *et al.*, 2008]. To sustain learning - one of dimensions of agility - under these conditions, organizations must "intentionally re-introduce variance into mature processes". Adler *et al.* [2009] summarize this as *deliberate perturbation.*

Deliberate perturbation is a theory that states that by sacrificing some efficiency in the short term, organizations may be able to sustain innovation and adaptability in the long run. If innovation yields increased efficiency in subsequent periods, then deliberate perturbation may be a dominant strategy leading to both higher efficiency and higher adaptability [Fine and Porteus, 1987]. The productivity challenge is to maintain appropriate performance in this kind of dynamic environment [Helo, 2004].

Helo [2004] provides a possible direction to answer our question regarding productivity in the agility context. He claims that there might be some trade-off between flexibility and efficiency, *but productivity depends on both.* Good efficiency may be productive in the short term, but in the case of product changes, the efficiency will be lost if the flexibility is low. For instance, this kind of situation may occur when inflexible software cannot cope with model changes. On the other hand, great flexibility is not productive if the software development is very repetitive. Helo [2004] describes organizations in the first stage, in which company maximizes the performance by concentrating on innovation, flexibility and responsiveness. Later, the company operates with growing capacity. In the last phase, cost minimization is important, in which measures such as cost per unit, labour productivities, etc., are typically used. This cycle is also known as product life cycle in the Marketing field. Figure 2.1 illustrates these phases, and their relationship with the company's product development and establishment.

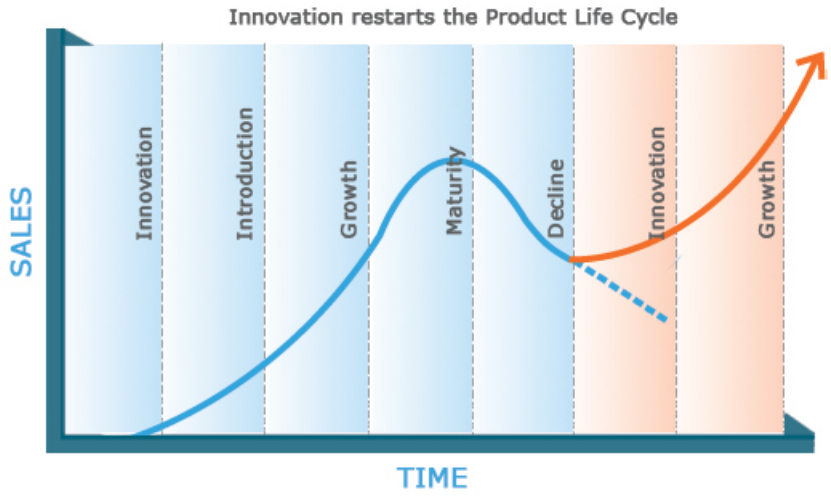

Figure 2.1: *Product life cycle (adapted from [Utterback and Abernathy, 1975]*

[Helo, 2004] also argues that agility can be seen as the capability to make a profit by keeping productivity level high in a changing environment. He presents agility as a short-time measurement,

defining company's agility as the sensitivity of the productivity. Thus, even appearing to be contradictory or conflicting, agility and productivity are positively associated. In our view, this is a paradox that should be understood in order to properly address productivity in agile environments. Figure 2.2 illustrates how flexibility (a dimension of agility) and efficiency (a dimension of productivity) are independent components of productivity. Volume levels (orders volume) and product mix (orders variety) may fluctuate as life cycles change.

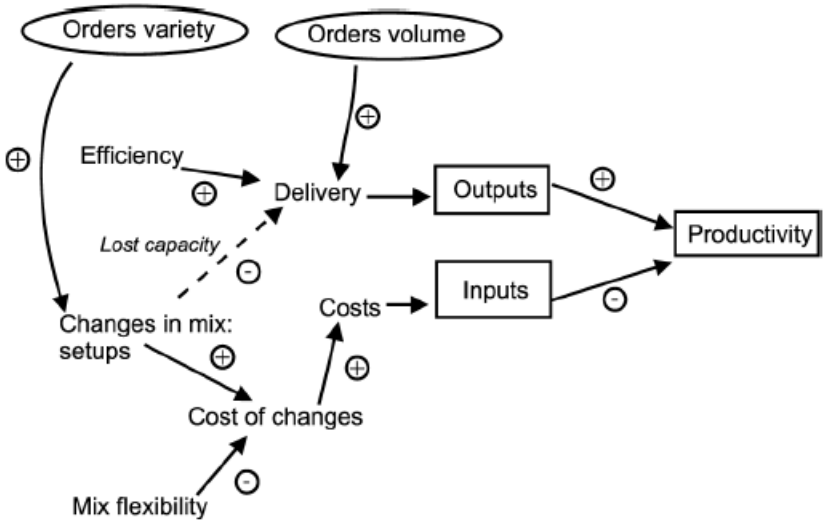

Figure 2.2: *Efficiency and flexibility are independent components in productivity [Helo, 2004]*

Adler *et al.* [2009] explain that a *tension between efficiency and flexibility/adaptability exists in most organizations*. Efficiency is observable almost immediately, while adaptability manifests itself over years or decades. Organizational survival depends on successfully managing this set of conflicting goals. No simple solution has yet been found for managing those contradictory goals, but much is known about the nature of such tension and possible approaches to handling it. This is, therefore, a challenge for this thesis: to grasp the nature of this tension and develop strategies to cope with productivity in agile teams.

2.5 Summary and the Challenges of this Thesis

This chapter has discussed the agile software development genesis and its main theoretical underlying concepts contrasting with traditional approaches. We then presented the agility concept based on a literature review, a concept not yet well delimited. Finally, we contrast agility and productivity through a paradoxical view, adopting the perspective in which productivity depends on both agility and efficiency.

Some challenges in agile software productivity were discussed in the previous sections. This section describes which of these challenges are the subjects of this thesis in the context of agile team productivity evaluation. This Thesis Research Challenges (RCs) can be defined as:

RC1. Working with the non-well established concept of *agility*. The issue here is to deeply explore and evaluate current definitions and verify how the studied companies fit into these definitions. Any empirical study on agile team productivity should start from the agility concept adopted by the team, and try to harmonize it with the theoretical assumptions.

RC2. Grasp the paradoxical relationship between agility and productivity. There is little (both theoretical and empirical) research on paradoxes when adopting agile methods, in particular regarding agility and productivity. While agile literature tends to take a simplistic view on the tension between software development agility and development performance, there

is a need to understand how to balance them in short and long term perspectives. The challenge is to understand the tension and develop strategies to cope with productivity evaluation in agile teams.

RC3. Understand to which extent productivity is important for agile teams. Considering the focus on agility and adaptability, we need to investigate how important is productivity for companies adopting agile methods. In this case, we should approach as many companies as we can to provide statistical significance.

Chapter 3

Software Development Productivity

Software development productivity is a general strategic concern in several industrial sectors [Baskerville *et al.*, 2006]. Although productivity has been studied extensively, it remains a controversial issue [Trendowicz and Münch, 2009]. First, several concepts are involved in its definition, including effectiveness, efficiency, performance, generating misunderstandings, and term overload [Trendowicz and Münch, 2009]. Second, the meaning of productivity varies according to the context [Tangen, 2005] and perspective [Petersen, 2011]. For instance, organizational productivity goals depend on project, individual, or task-specific productivity goals. Each level needs its own special set of standards, norms, measures, and metrics to enable effective management at that level [Dale and van der Zee, 1992].

Finally, there is no consensus on the best way to measure software productivity, both in traditional and agile software development teams, because software development is a human-based activity with extreme uncertainties from the outset, leading to many difficulties in achieving a reliable software productivity definition [Trendowicz and Münch, 2009]. The diversity of these aspects hinders any precise approach to define and measure software productivity.

In this chapter, we give an overview of software productivity definitions and their relationship with each other, in pursuit of a more accurate meaning of productivity in the current context of software development. We also summarize general software productivity metrics, measurement and factors. Then, we introduce team productivity monitoring in dynamic environments. All these definitions are important to properly define agile team productivity.

Throughout the chapter, we outline propositions based on the theory that will be useful to better delineate research goals and analyze empirical data. Kitchenham [2010] has noticed, for instance, that many case studies in software measurement do not present detailed propositions before data analysis takes place. We thus enumerate some important propositions that will be used in the next chapters.

3.1 Productivity definitions

As mentioned, productivity is misunderstood because it is often considered to be interchangeable, along with terms such as efficiency, effectiveness, and profitability. Indeed, productivity is a multidimensional concept. After an extensive review of productivity definitions, Tangen [2005] has proposed a model to illustrate the relationship between productivity, effectiveness, efficiency, profitability, and performance, the Triple-P model (Figure 3.1). The model aims to clarify the different scopes related to productivity and other similar concepts.

The Triple-P model defines *Productivity* as the relation between output quantity (i.e., correctly produced products that fulfill their specifications) and input quantity (i.e., all resources that are consumed in the transformation process). The next level is represented by *Profitability*, a monetary input/output relationship where price factors are included. Profitability is commonly calculated by the ratio between *Revenue* and *Cost*, or simply by the *Return of Investiment* (ROI). *Performance* is the "umbrella term of excellence" that covers profitability, productivity, and non-cost factors such as

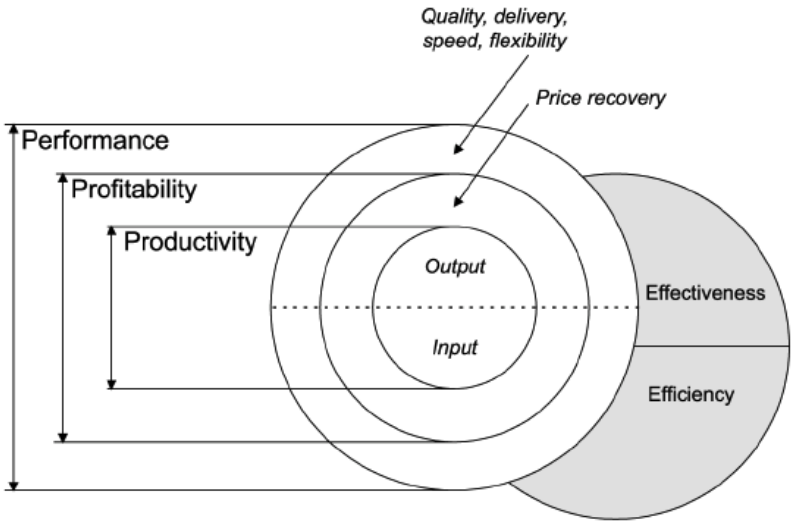

Figure 3.1: *Triple-P model: Productivity and related concepts from [Tangen, 2005]*

quality, speed, delivery, and flexibility. Effectiveness and efficiency are considered cross-functional in this model. *Effectiveness* represents the degree to which desired results are achieved; *Efficiency* represents how well the resources of the transformation process are utilized. Thus, effectiveness and efficiency occurs in all three levels: productivity, profitability, and performance. It is possible to notice efficiency and effectiveness dimensions, for instance, in the productivity definition: the capacity of producing quality outputs (effectiveness) with the best utilization of input resources (efficiency).

However, software development is mental work involving knowledge creation or, at least, knowledge used as a dominant part of the work [Ramírez and Nembhard, 2004]. A Knowledge Worker (KW) has been described in the literature as a high-level employee who applies theoretical and analytical knowledge, acquired via formal education and experience, to develop new products or services [Drucker, 1994]. The product of a knowledge worker is typically intangible: knowledge is the addition of meaning, context and relationships to data or information [Bosch-Sijtsema *et al.*, 2009].

Knowledge Worker Productivity (KWP) is strongly related to the degree of *autonomy*, *responsibility*, and *continuous life-long learning* a person experiences. According to the extensive review conducted by Ramírez and Nembhard [2004], knowledge worker productivity is also related to the customer satisfaction, innovation/creativity, timeliness, product quality, profitability, and team efficiency and effectiveness. Table 3.1 describes the dimensions we consider more closely related to software development, based on Drucker's [Drucker, 1994, 1999] and Ramírez and Nembhard's work [Ramírez and Nembhard, 2004].

The definition of KW productivity goes beyond the Triple-P Model by including people factors. KW productivity encompasses many dimensions from productivity, profitability, and performance, also adding new dimensions such as autonomy, responsibility, learning, and creativity. Thus, these new dimensions are also a sign of productivity increasing in software development individuals and teams. Figure 3.2 illustrates the relationship between KW productivity and the Triple-P Model dimensions.

The productivity definition is frequently associated to metrics. It is common to find studies that adopt definitions based on formulas, instead of first discussing productivity theory and its many dimensions. The next section presents software productivity measurement and review approaches to understand productivity through metrics.

Table 3.1: *Knowledge worker (KW) productivity dimensions*

KW Productivity Dimensions	Description (*adapted from Drucker [1994, 1999], Ramírez and Nembhard [2004]*)
1. Quantity	Accounts for outputs (quantities) and outcomes (quantification of qualitative variables such as customer and worker satisfaction)
2. Costs & profitability	Accounts for profitability, costs, etc.
3. Timeliness	Accounts for meeting deadlines, overtime needed to complete the work, and other time related issues.
4. Autonomy	Accounts for independence and how many things a worker can do simultaneously.
5. Efficiency	Accounts for "doing things right". Refers to any task, even if it is not important for the job. The task is completed meeting all the standards of time, quality, etc.
6. Quality	Accounts for how good the work is.
7. Effectiveness	Accounts for "doing the right things". Refers just to the tasks that are important for the job, even if they are completed without meeting standards of time, quality, etc.
8. Project success	Accounts for overall result of work, considering decision-making, team interaction, communication, predictability, crisis management, documentation, transferability of work, etc.
9. Customer satisfaction	Accounts for the fact that the product needs to add value to the customer's business.
10. Innovation & creativity	Accounts for the ability of creating new ideas to improve productivity.
11. Responsibility & importance of work	Accounts for the importance of performing well at critical times.
12. Learning	Knowledge work requires continuous learning and teaching.

3.2 Software productivity measurement

Software measurement is widely recognized as an effective means to understand, monitor, control, predict, and improve software development and maintenance projects [Briand *et al.*, 1996]. Practical application of engineering, including software development, in the industry would not have been possible without measurement, which allows and supports production planning, monitoring and control, decision making, cost/benefit analysis (especially when new techniques are proposed or introduced), post-mortem analysis of projects, and learning from experience [Morasca, 2001].

In a systematic review on software productivity measurement, Petersen [2011] identified two measurement purposes: *reactive* and *predictive*. *Reactive* is to determine the productivity based on data of the past. *Predictive* is the attempt of quantifying the productivity expected in the future. Predictions of productivity are useful to determine whether corrective actions are needed, and it can be combined with reactive measurement to verify if past productivity is a trend that is likely to continue.

Measuring software productivity is often a software process improvement activity [Petersen, 2011, Trendowicz and Münch, 2009]. The famous quote "You cannot predict nor control what you cannot measure" [Fenton and Pfleeger, 1997], indeed, has influenced many organizations to adopt

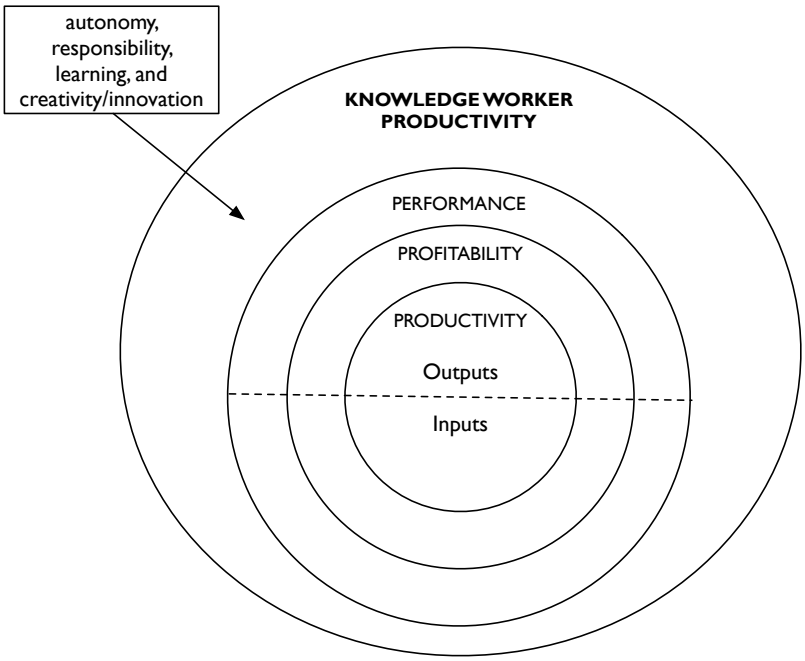

Figure 3.2: *Relationship between Knowledge worker productivity and the Triple-P Model*

productivity metrics to define a *baseline* for improvement.

Productivity baselines, in general, are based on hard, quantitative data from many productivity dimensions, such as productivity rates of different types of software projects (divided by applications, programming languages, etc.), management and staff factors such as motivation and specialization availability, methodology factors (methods, tools, and procedures adopted) etc. All these measurement initiatives are often part of an organizational measurement program, one important step for measurement-based process improvement [Jones, 2008, Morasca, 2001]. However, as performance has both *objective* and *subjective* measures [Na *et al.*, 2007, Sudhakar *et al.*, 2011], baselines can be also composed of subjective measures.

Productivity is a popular attribute in process measurement, and may have different facets depending on the process phase in which it is applied [Morasca, 2001]. In fact, productivity is a dynamic measure by nature and may differ with th products produced and the technology used [Bititci *et al.*, 2011]. In the coding phase, one may interpret productivity in terms of the amount of product or functionality delivered per time unit; in the verification phase, productivity may be interpreted in terms of the failures caused per time unit [Morasca, 2001].

Richardson and Gordon (1980) claim that companies need to utilise different performance measures at different stages of a product life cycle. In the first stage, the organization maximises the performance by concentrating on innovation, flexibility, and responsiveness. Afterwards, the organization operates with growing capacity. Appropriate measures in this stage include capacity utilisation, growth, backlogs etc. In the last phase, cost minimisation is important. Measures such as cost per unit, labour productivities, etc. are typically used. Richardson and Gordon (1980) also claimed that as product life cycles decrease, the importance shifts from productivity to measures related to innovation and flexibility. Thus, the understanding of appropriate measures in dynamic situations is essential [Bititci *et al.*, 2011].

Software productivity should be measured by programmers (self reports), project or team manager, outside analysts or observers, or automated performance monitors [Scacchi, 1994]. It should not only be measured at project completion, but also throughout the project in terms of how the team is doing [Pries-Heje and Commisso, 2010].

3.2.1 Why measure software productivity

According to Scacchi [1994], a number of reasons for measuring software productivity have been observed. As we mentioned, the main purpose is to identify opportunities for improvement, such as reducing software development costs, improving software quality, and improving the rate at which software is developed. Scacchi [1994] enumerates some possible advantages:

- increase the volume of work successfully accomplished by current staff effort,
- accomplish the same volume of work with a smaller staff,
- develop products of greater complexity or market value with the same staff workload,
- avoid hiring additional staff to increase workload,
- rationalize higher levels of capital-to-staff investment,
- reduce error densities in delivered products, and decreasing the amount of time and effort needed to rectify software errors,
- streamline or downsize software production operations,
- identify possible product defects earlier in development,
- identify resource utilization patterns to discover production bottlenecks and underutilized resources,
- identify high-output or responsive personnel to receive rewards, and
- identify low-output personnel for additional training or reassignment.

It is not convenient to try to accomplish most or all of these objectives through a single productivity measurement program. Each goal has its specific data, increasing the volume of data to collect, which can lead to inaccurate or misleading measurement [Kitchenham *et al.*, 2007]. Thus, a productivity measurement program must be carefully designed to avoid creating conflicts [Scacchi, 1994].

Intended uses of performance measurement in organizational settings can be usefully divided into two categories [Austin, 1996]:

Motivational measurement is explicitly intended to affect the people who are being measured. An example of measurement in this category is sales tracking linked to a sales commission system. Used in this way, measurement is an attempt to control individual activity, which it is assumed that will not be congruent with organizational objectives, absent the measurement. This is the use of measurement that is implicit in most conventional measurement frameworks.

Informational measurement is valued primarily for the logistical, status, and research information it conveys, which provides insights, support organizational learning, and allows better short-term management and long-term improvement of organizational processes. An example of measurement in this category is data gathering for the purpose of understanding how to redesign a business process. This use of measurement has little to do with control and much to do with learning. There are two types of informational measures: *process refinement* and *coordination.*The first one aims to reveal the detailed structure of organization processes. The last aims to provide real-time information that enable short-term management, such as flows and schedules.

Austin [1996] states that, in knowledge work settings, *motivational* uses of measurement are unlikely to be helpful due to the distortions generated by the information incompleteness, so-called *measurement dysfunction.* For instance, controlling individual performance through targets has the

risk of staff members manipulating the system to their benefit and the expense of other teams and even the entire organization [Brudan, 2010].

In the same direction, Deming [1986] has recommended 14 points for management, one of them is: "eliminate numerical goals for people in management". He emphasizes the need of learning about the customer needs, instead of managing by numbers, to achieve continuous improvement of quality and productivity. He advises to eliminate standards that prescribe quotas for the work force and numerical goals for people in management [Morasca, 2001].

Recent research also suggest that the ultimate goal of performance measurement should be *learning*, *improvement*, and *goal achievement* rather than control [Bititci *et al.*, 2011, Brudan, 2010, Davenport and Harris, 2007]. In this way, it would be possible to use measures avoiding dysfunctional behaviors. Brudan [2010] contrasts measurement for control with measurement for learning, as shown in Table 3.2.

Table 3.2: *Measurement for control compared to measurement for learning [Brudan, 2010]*

Characteristic	Measurement for control	Measurement for learning
Measurement drivers	Management	Employees
Measures development	Top-down commands	Process-oriented bottom-up approach
Measurement role	Measuring and managing work in functional activities.	Measuring and managing the flow of work through the system
Measurement focus	Productivity output, targets, standards: related to budget	Capability, variation: related to purpose
Results communication	Restricted	Open
Target driven by	Budget/political aspirations	Understanding achievement versus purpose
Follow-up to results	Rewards, punishment and action to improve results	Dialogue and improvement
Learning cycle	Single loop	Double loop learning
Link to rewards	Link to individual rewards and recognition system	Group rewards, based on improvement

Traditionally, organizational performance management has been concerned with control, by setting and monitoring achievement of targets at all organizational levels. Command and control style considers organizations as top-down hierarchies, where managers make decisions using budgets, standards, and targets. Workers are controlled with a variety of management tools and practices (rules, specifications, procedures, inspections, performance management reviews) [Brudan, 2010]. In this context, measurement is used to generate information to keep things under control, or reinforce control within employees. The basic definition of productivity (output/input) and other budget targets are the measurement focus. Considering rewards systems, this approach links measurement to individuals and its own performance.

A shift was proposed by the knowledge management school of thought, that argues that it is not possible to accurately measure social phenomena. Some researchers, such as DeMarco, recently have recognized that his own statement "you can't control what you can't measure" (from [DeMarco, 1986]) should not be strictly followed [DeMarco, 2009]. He illustrates this new way of thinking with successful projects such as GoogleEarth or Wikipedia, which have succeeded without much control. Measurement for learning and improving appeared as an alternative approach to apply metrics, having roots in Knowledge Management and Systems Thinking [Brudan, 2010]. It emphasizes a process-oriented bottom-up approach of measurement development, driven by employees. The measurement role is managing the flow, not functional activities. This shift represents a major focus on the final outcome, not on the partial result delivered by each employee. The target is to understand achievement versus purpose, instead of controlling the budget. Finally, measurement for learning makes a room for dialogue and improvement, recognizing primarily the group performance.

Proposition 1: Productivity measurement should be used predominantly for learning, short-term management, and long-term improvement. As it is a dynamic measure by nature, it may differ with the products and the technology used. The set of metrics may vary at different stages of a product life cycle and should be developed in a process-oriented, bottom-up approach.

3.2.2 Traditional software productivity metrics

In a recent systematic review, Petersen [2011] identified eight measurement-based approaches to analyze software productivity. They divided them into abstraction categories, such as task, individual, project, process, and organization levels. Most approaches found were studied at the project level. All approaches were designed to traditional software development paradigms. The main approaches presented by Petersen [2011] are:

Weighted productivity factors. Productivity is calculated by weighting factors influencing the productivity. A common approach to identify weights of independent variables to determine a dependent variable is regression analysis.

Simple Input/Output Ratios. Are based on the classical productivity model, defined as the ratio of one output divided by one input. The model can be extended by adding up several inputs as well as outputs.

Data envelopment analysis (DEA). Is able to handle multiple inputs and outputs. The inputs and outputs are weighted and their ratio (productivity) should be maximized for each decision-making unit. The weights are determined by the method. DEA also allows identifying efficient reference projects for each inefficient projects.

Bayesian belief networks. Productivity can be defined in intervals of productivity values. Based on the probabilities assigned to the variables and the knowledge of cause-effect relationship the probability of achieving a specific productivity can be calculated.

Earned value analysis. Is the output computed as the percentage of progress towards the final product. The progress can only be measured with clearly defined milestones. The calculation shows the shifts in productivity and the progress.

Statistical process control. Uses statistical inferences and control charts to analyze the software process. If data points are outside the control limits, then this indicates that the process is out of control.

Balanced scorecard. It is a tool to evaluate an organization based on a long-term perspective (strategies and visions). The scorecard includes different perspectives that are evaluated (e.g., finances, human and management aspects, process perspective, and customer satisfaction). The measures obtained are compared to the target levels defined based on the goals derived from strategies and visions.

Metric space. Allows measuring the distance between elements of a set or between functions. For example, to compare the productivity of two programmers the distance between their accumulated work functions over time could be calculated.

We illustrate the productivity measurement approaches and examples of measurements in Table 3.3. The systematic literature review did not mention knowledge worker productivity measurement. Most studies included in the systematic review have been published before 2000, and hence do not take recent developments in software engineering research and practice into consideration - for instance, agile methods are not considered.

Petersen [2011] argues that it is not possible to say whether productivity measurement and prediction will be harder or easier in the agile context. He emphasizes that the change of the ways

Table 3.3: *Measurement-based approaches to analyze software productivity (adapted from [Petersen, 2011])*

Measurement approach	Example of metrics
Weighted productivity factors	simple linear regression, non-linear regression, multiple regression, and stepwise regression
Simple Input/Output Ratios	Outputs: Lines of code (LOC) [Fenton and Pfleeger, 1997] or Function points (FP) [Maurer and Martel, 2001]. The most common input is measured as effort used to develop the output.
Data envelopment analysis (DEA):	-
Bayesian belief networks	-
Earned value analysis	EVA [Kadary, 1992], Return of investment (ROI) [Fowler, 2003], Revenue per employee [Poppendieck and Poppendieck, 2003]. Input is the effort spent from the beginning of the project till the completion of a milestone. The productivity is the ratio of the earned value divided by the milestone cost.
Statistical process control	Time-series analysis, Predicted vs. actual: actual values close to lower prediction interval
Balanced scorecard	Specific for each company
Metric space	Distance between accumulated work of 2 programmers.

of working in agile projects needs to be considered when developing prediction and measurement approaches. He also recognizes that value-based metrics were not well studied. In the value-based approach, the target would not be to produce as much as possible with as little effort as possible, but *produce as much value as possible with as little effort as possible*. This also relates to the agile methods principles. However, value-based metrics are still an under-researched topic.

Thus, we identify two research needs based on Petersen's results: i) there is a need for studies in productivity metrics in the agile context, since most studied metrics only fit into traditional approaches and ii) agile productivity metrics should be defined in a different way, considering also a value-based approach. Our Research Question 3 (RQ3) focuses on the first research need.

Productivity metrics scope

Productivity definitions may vary according to the perspective [Petersen, 2011], often causing confusion. We classified the most popular productivity metrics (not necessarily the most studied ones) using the Triple-P model [Tangen, 2005]. Table 3.4 shows some software productivity metrics, their classification and key references.

While some metrics, such as Function Points or LOC, are more basic metrics, others such as Return of investment (ROI) refer to a broader perspective, such as profitability or performance (and nevertheless are denominated productivity metrics by many researchers).

Some authors [Fenton and Pfleeger, 1997, Moser *et al.*, 2007] consider that software productivity is the ratio between lines of code (LOC) produced and the effort in hours to produce them. However, LOC is a limited metric in the presence of different programming languages [Jones, 2008], often resulting in misleading measurement [Kitchenham *et al.*, 2007]. For Maurer and Martel [2001], functionality rather is expressed in Function Points. Yet, in the case of measurement of heterogeneous projects with unstable processes, using these simple productivity metrics may lead to serious problems. In consequence, organizations that failed to measure productivity using simple metrics find it either difficult to interpret and/or perceived little benefit [Trendowicz and Münch, 2009].

Table 3.4: *Software productivity metrics, scope and main references*

Definition	**Scope** (based on [Tangen, 2005])	**Reference**
Lines of Codes (LOC)/hours	Productivity	[Fenton and Pfleeger, 1997]
Function points/hours	Productivity	[Maurer and Martel, 2001]
Return of investment (ROI)	Profitability	[Fowler, 2003]
Revenue generated per employee	Profitability	[Poppendieck and Poppendieck, 2003]
Value added/input of production factors	Profitability	[Hartmann and Dymond, 2006]
Knowledge worker (KW) productivity is not only a matter of quantity of output, quality is at least as important	Performance + People factors (e.g., learning and creativity)	[Drucker, 1999, Ramírez and Nembhard, 2004]

Agile methods encourage businesses to be accountable for the value produced by software development efforts [Hartmann and Dymond, 2006]. Value is defined as software put into production that can return an investment over time. The more rapidly high-value software can be rolled out, the quicker its value is realized. There are many ways to measure value and, consequently, measure agile team productivity on delivering value. Poppendieck and Poppendieck [2003] recommend that companies that produce and resell software consider productivity as revenue generated per employee. Fowler [2003] states that productivity of teams can only be really measured if there is a Return of Investment (ROI) evaluation provided by the delivered software. These value-based productivity definitions imply that the target would not be producing as much as possible with as little effort as possible, but producing as much value as possible with as little effort as possible [Petersen, 2011]. Thus, we can classify these approaches within the Profitability category of the Triple-P model, since they rely on profit measurement as a way to understand productivity.

As mentioned before, KW Productivity goes beyond the Triple-P Model. Even if there are no universally accepted methods to measure knowledge worker productivity, it has been studied in terms of productivity dimensions such as KW's perception of their productivity, customer satisfaction, quantity of work, innovation, creativity, timeliness, product quality, absenteeism, profitability, and team efficiency and effectiveness, as presented in Section 3.1.

Consonant with Drucker's (1994) definition of knowledge worker productivity, we argue for:

> *Proposition 2:* Software development productivity should be evaluated in a holistic way, since it involves different perspectives, human knowledge, learning, creativity, as well as performance and profitability.

3.2.3 Performance measurement systems

Performance can be measured at different levels in an organization. It may be suitable at the company or business unit level to support strategic or long-term decision-making processes by senior management. Or performance can be measured at the "shop-floor" level to support daily or short-term decision making processes by local management or operating staff. In general, different dimensions of performance should be controllable at different organizational levels [de Haas and Kleingeld, 1999].

A performance indicator is a formula or rule that enables quantification of performance. Quantification is the essence of measurement: the adding of symbols (i.e., figures) to phenomena (i.e., performance) through a set of prescribed rules (i.e., indicators). A set of performance indicators

with procedures for periodic data gathering and the group of organizational actors they relate to, form the elements of a Performance Measurement System (PMS) [de Haas and Kleingeld, 1999].

Productivity is often one of the indicators in performance literature, as well as quality, delivery, cycle time, waste, customer satisfaction, flexibility, etc. [Lynch and Cross, 1991]. As discussed in Section 3.1, performance also can bee seen as knowledge worker productivity, since both concepts have many dimensions in common. Therefore, we assume PMS as a synonym of knowledge workers' measurement system. In this sense, PMS literature becomes relevant for our research, because it can help us to understand how to develop and adapt a productivity measurement system for knowledge workers over time.

Developing and updating Performance Measurement Systems

The turbulent nature of the organizations operating environment leads to the need to understand how performance-measurement systems (PMS) can be used and how they could adapt to the changing operating environment [Bititci *et al.*, 2011, Mintzberg, 1987]. However, the rate and scale of change are increasing in the last decades. In the beginning, change was slow and incremental in the environment. Nowadays, they are turbulent and discontinuous. Thus, it is challenging to develop a PMS that "evolves in response to changes in the organizations inner and outer operating environment" [Bititci *et al.*, 2011]

Choosing the right indicators for individuals, groups, and organizations is part of the PMS development. Creating and deploying a PMS involve many activities into different phases. Bourne *et al.* [2000] provide a comprehensive overview of how performance-measurement systems evolve. They suggest four main phases to develop performance measurement systems, as shown in Figure 3.3.

The design phase can be subdivided into identifying the key objectives to be measured and designing the metrics themselves.

The implementation of the performance measures is the phase in which systems and procedures are put in place to collect and process the data that enable the measurements to be made regularly. It may involve initiating new procedures, so that information currently not recorded is captured and it may involve completely new initiatives, such as the setting up of a regular customer or employee survey.

The use of the performance measures is split into two main subdivisions. First, as the measures are derived from strategy, the initial use to which they should be put is that of measuring the success of the implementation of that strategy. Second, the information and feedback from the measures should be used to challenge the assumptions and test the validity of the strategy

The performance measurement system should be continuously updated and requires developing and reviewing at a number of different levels as the situation changes [Bourne *et al.*, 2000]. It should include:

- an effective mechanism for reviewing and revising targets and standards;
- a process for developing individual measures as performance and circumstances change;
- a process for periodically reviewing and revising the complete set of measures in use. This should be done to coincide with changes in either the competitive environment or strategic direction, and should include an effective mechanism for reviewing and revising targets and standards

Implementation of the individual measures does not create a performance measurement system. Measuring is only one part of using the measures. A forum is needed to review the measures and ideally to achieve agreement on actions. To do this, a regular meeting is required, and it must be

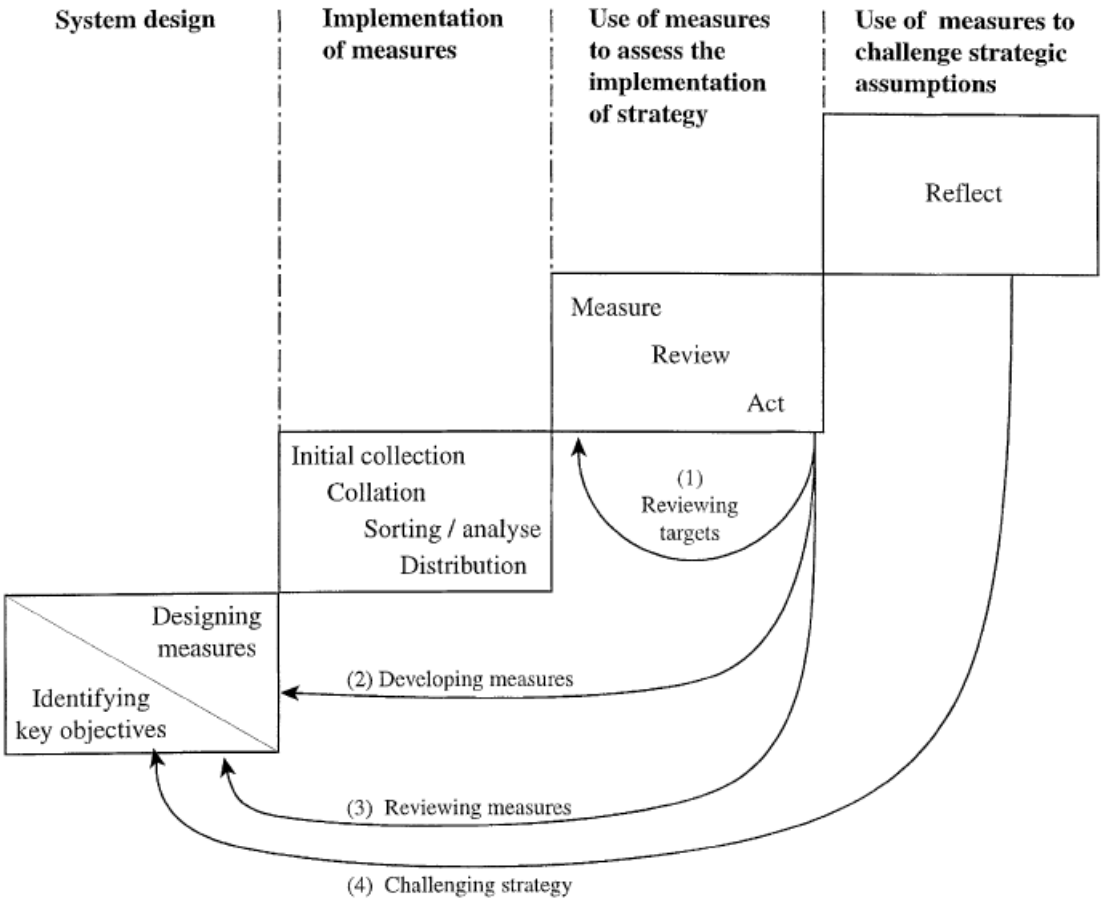

Figure 3.3: *Phases in developing performance measurement systems [Bourne et al., 2000]*

attended by organizational members who have responsibility for the performance being measured. In addition, these phases can overlap and some measures can be implemented before all the measures have been completely designed [Bourne *et al.*, 2000].

We can understand PMS development as an *incremental process* in organizations. We argue that PMS development is compatible with IT companies using software development approaches that evolve and improve incrementally, such as Agile methods. Thus, it would be possible to develop a PMS within agile teams without generating additional overhead, since metrics can be chosen, integrated, and analyzed incrementally, as it is normally done in an agile project. For instance, through the XP tracker role [Beck and Andres, 2004] .

Based on the review presented by Bititci *et al.* [2011], there are three grand challenges for PMSs in research and practice. Two of them are directly related to our research: i) understanding performance measurement as a social system and ii) understanding performance measurement as a learning system. Bititci *et al.* [2011] state that there is a shift in purpose from *rational control* towards *cultural control* and *learning*. This is aligned with our vision that measurement should be used to learn, particularly in agile teams.

There is also an epistemological shift from positivism towards interpretivism in performance measurement systems [Bititci *et al.*, 2011]. This shift is reflected in the research method choice for Research Question 3 ("How to monitor productivity factors, considering agility and adaptability? How do agile metrics support productivity monitoring?"). We better discuss this issue in Chapter 7.

Proposition 3: It is possible to develop a Performance Measurement System within agile teams without generating additional overhead, since metrics can be chosen, integrated, and analyzed incrementally, as it is normally done in an agile project.

3.3 Software productivity factors

We identified three literature reviews [Sampaio *et al.*, 2010, Trendowicz and Münch, 2009, Wagner and Ruhe, 2008] on productivity factors in software engineering. Based on these three studies, we located the most relevant sources and included, in our literature review, only those founded on some empirical work studying the effect of the factor on productivity. We did not include studies that do not provide novel findings on productivity factors, such as work that use COCOMO factors without any new insight, or essays discussing productivity factors without any empirical evidence. Moreover, as we are considering team productivity, we will focus more on factors influencing team productivity.

3.3.1 Product factors

Product factors are related to a specific characterization of software, such as domain, requirements, architecture, code, documentation, interface, size, etc. Table 3.5 presents the results of our literature research on product factors influencing software productivity.

Table 3.5: *Product Factors influencing software productivity*

Productivity factors	Positive impact	Negative impact	Neutral
Reuse of software products, processes, and artifacts, including components, frameworks, and software product lines.	[Atkins *et al.*, 2000, Basili *et al.*, 1996, Lewis *et al.*, 1991, Lim, 1994, Morisio *et al.*, 1999, Siy and Mockus, 1999]		[Frakes and Succi, 2001]
Software characteristics, such as architecture, complexity, domain, non-functional requirements, stability requirements, user interface, and software size.	[Maxwell *et al.*, 1996, Tan *et al.*, 2009]	[Boehm, 2000, Krishnan, 1998, Tan *et al.*, 2009, Vosburgh *et al.*, 1984]	[Fenton and Pfleeger, 1997]

Reuse. Several studies have found that reuse impacts positively on the productivity. Lewis *et al.* [1991] state that software reuse improves productivity no matter which language paradigm is used. Lim [1994] conducted a large study on the effects of reuse on the economy, quality, and productivity of Hewlett-Packard (HP) projects. Reuse reduced the time-to-market and increased productivity in 57%. Basili *et al.* [1996] show a strong impact of reuse on product productivity and especially on the product quality, considering defect and rework density. Furthermore, when developers are more familiar with generic software, such as frameworks, the reuse benefits are even greater. The observed productivity gain of each development where the asset is reused reached 280%, and the benefit from creating and maintaining reusable assets increases with the number of its reuses [Morisio *et al.*, 1999].

More recently, domain engineering and software product lines should be considered as a potential factor influencing productivity [Trendowicz and Münch, 2009]. Atkins *et al.* [2000] conducted an empirical study where the change effort of large telecommunication software was reduced by about four times when the domain-engineering technologies were applied. Similarly, Siy and Mockus [1999]

found that the change effort is reduced three to four times in their example (in line with generally accepted view that Domain Engineering techniques improves software productivity two to ten times). However, in a quasi-experiment using data from industrial organizations, Frakes and Succi [2001] found that data were inconsistent regarding the relationship between reuse and productivity with some relationships being positive and others negative, warranting further study.

Software characteristics. According to Tan *et al.* [2009], software architecture drives productivity variation, since a well-planned and organized architecture tends to increase productivity. On the other hand, a deficient architecture will result in productivity decrease because of the needed re-architecting to bring the project back on track.

Vosburgh *et al.* [1984] found that productivity decreases with higher percentage of complex code. A more specific research concluded that complex projects require more effort, coordination, and create loss of efficiency, which results in lower productivity [Boehm, 2000]. Fenton and Pfleeger [1997] reported a different result: software complexity might have a positive impact on productivity in the context of new development and a negative one in case of maintenance.

According to the review presented by Trendowicz and Münch [2009], the application domain is considered as the second most significant cross-context factor (i.e., a factor that describes specific productivity modeling contexts). In this sense, Putnam and Myers [1996] found out that productivity varies across domains and provides, for instance, evidence that projects in a real-time domain tend to be less productive, than in the business systems domain.

In two different studies, Maxwell shows that non-functional requirements, such as efficiency and reliability, tend to impact on productivity negatively [Maxwell and Forselius, 2000, Maxwell *et al.*, 1996]. Considering requirements stability, researchers agree that stable requirements impact positively on software productivity [Maxwell and Forselius, 2000, Vosburgh *et al.*, 1984].

Productivity also depends on the user interface [Maxwell and Forselius, 2000]. According to Maxwell and Forselius [2000], productivity is twice as high for projects with a graphical interface, rather than a textual one. The authors claim that graphical user-interface projects may have higher productivity because of the combination of effects of many factors' evolution over time (e.g., modern programming practices, tools).

Regarding the relationship between project productivity and system size, Vosburgh *et al.* [1984] found that productivity decreases as the number of the developed statements increases in a corporate-wide study of 44 IT programming projects in nine countries. Krishnan [1998] conducted a 2-year case study in a company that developed commercial software systems for various applications and found that the more the product size increases, the more field defects are found and the lower is the productivity. Conversely, Maxwell *et al.* [1996] found that productivity increases with increasing system size. Recently, Tan *et al.* [2009] presented a study on productivity trends in incremental and iterative software development. They found that, when the project size increases, integrating components and modules cause notable problems, and to resolve them, the team decided to re-architect which paid off in the following increments. But the authors state that re-architecting helps to maintain productivity, although the effect may be short-lived.

3.3.2 Personnel factors

Personnel factors involve team member capabilities, experience, and motivation. Table 3.6 presents the results of our literature research on personnel factors influencing software productivity.

Team experience and capabilities. Traditionally, team capabilities and skills are the most significant personnel characteristics that influence productivity, generally understood as a moderator when studying factors impacting productivity [Blackburn *et al.*, 2000, Trendowicz and Münch, 2009]. Banker *et al.* [1991] found that team capability and application experience explain significantly software maintenance productivity. Krishnan's case study [Krishnan, 1998] indicates that personnel capability of the team is significantly associated with higher quality and increasing productivity,

Table 3.6: *Personnel Factors influencing software productivity*

Productivity factors	Positive impact	Negative impact	Neutral
Team experience and capabilities including customer experience, domain knowledge and experience, generational experience, i.e., percentage of the development team already participating in two or more generations of software projects, programming language experience, staff capabilities and experience, and experience with tools.	[Blackburn *et al.*, 1996, Krishnan, 1998, MacCormack *et al.*, 2001, Maxwell and Forselius, 2000, Maxwell *et al.*, 1996, Tan *et al.*, 2009, Vosburgh *et al.*, 1984]		[Faraj and Sproull, 2000]
Motivation to work on the project and in the company.	[Beecham *et al.*, 2008, Boehm, 1981, Procaccino *et al.*, 2005, Sharp *et al.*, 2009]		
Project Management including aspects of quality of management, conflict management, task assignment, and administrative and formal coordination.	[Blackburn *et al.*, 2000, 1996, Boehm, 2000, Faraj and Sproull, 2000, Maxwell *et al.*, 1996, Sawyer and Guinan, 1998]		[Smith *et al.*, 2001]

as well as teams with higher experience on specific application domain implies lower maintenance costs.

Maxwell and Forselius [2000] analyzed data of 26 companies in Finland and found that productivity increases with the staff's increasing tool capabilities. MacCormack *et al.* [2001] used the "generational experience" concept referring to the percentage of the development team that already participated of two or more generations of software projects. He concluded that generational experience explains 24% of the resource productivity variation. Tan *et al.* [2009] also state that staff capabilities and experience are influential factors, since there is a relationship with increased confidence, and reduced implementation level code breakage. Vosburgh *et al.* [1984] highlighted the positive influence of customer experience in software productivity. Blackburn *et al.* [1996] explain that developers from the fastest and more productive firms usually spend more time learning the customer needs and domain.

Maxwell *et al.* [1996] discovered that programming language experience has no significant effect on productivity. Faraj and Sproull [2000] also discuss that the professional experience may have no impact on team effectiveness, but only on efficiency. They argue that expertise is necessary, but its "mere presence" does not impact significantly on team effectiveness, unless team members coordinate their expertise.

Motivation. Productivity can be seen as an outcome of the software engineer's motivation [Beecham *et al.*, 2008]. However, due to the difficulty to measure it, there are few empirical studies showing its relationship with team productivity. Applying our inclusion criteria, we found two empirical studies on motivation and productivity. In the first study, Procaccino *et al.* [2005] conducted a survey and observed a relationship between practitioners' motivation and increasing productivity. The authors recommend that project managers should pay attention to task assignment, interpersonal relationships, work conditions, and environment to motivate work in software development.

The second study, conducted by Sharp *et al.* [2009], organizes the knowledge about motivation in Software Engineering. The authors present the MOCC model (Motivators, Outcomes, Characteristics, and Context), a model built from previous empirical evidence provided by the systematic

review on motivators in software engineering of Beecham *et al.* [2008]. Figure 3.4 presents the MOCC model. This work has done a dramatic contribution to the field, because over the last 30 years research on software engineer motivation has been isolated from classical motivation theory, leading to different nomenclatures and hampering the synthesis of existing findings.

Motivators can be intrinsic (derived from the pleasure of doing the work itself) or extrinsic (related to factors external to the job, such as working conditions) [Herzberg, 1993], as shown in Figure 3.4. The motivators inherent to software engineering are all intrinsic factors as they solely relate to software engineering [Sharp *et al.*, 2009]. Software engineer's characteristics guide the individual towards certain motivation factors, while motivators themselves influence the strength of individual characteristics. Characteristics of software engineers are influenced by contextual factors, most specifically the individual's personality and the environment in which they are practicing. Finally, there are many possible external signs or outcomes of motivated software engineers. Sharp *et al.* [2009] highlight retention, productivity, project delivery time, adherence to budgets, low absenteeism, and improved project success.

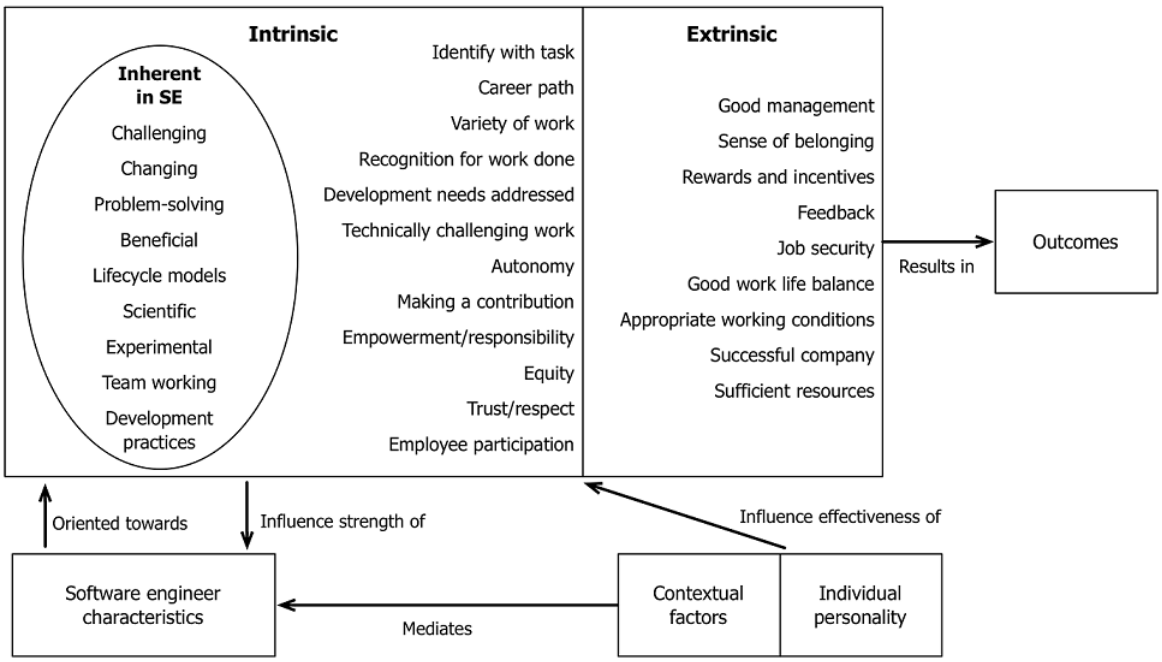

Figure 3.4: *Sharp's Motivators, Outcomes, Characteristics, and Context model*

3.3.3 Project factors

Project factors involve team member capabilities, experience, and motivation. Table 3.7 presents the results of our literature research on project factors influencing software productivity.

Some factors are related to specific project characteristics, such as project management, project duration, team allocation and size, communication, supportiveness, and coordination. Although some factors overlap between personnel or process factors, we organized all team factors in this section. For instance, Sawyer and Guinan [1998] consider formal and administrative coordination as production (i.e., process) factors, and informal communication and coordination as social process. We will address these social processes as project factors.

Most researchers agree that poor project management may affect negatively on the project productivity [Blackburn *et al.*, 2000, Boehm, 1981, Maxwell *et al.*, 1996]. Proper task assignment can raise productivity [Boehm, 1981], although Smith *et al.* [2001] discuss that it depends on task intensity, concurrency, and fragmentation. Conflict management is another dimension that explains productivity variations [Sawyer and Guinan, 1998]. It includes the ability to overcome

differences among team members and negotiate commitments effectively. Administrative and formal coordination may impact on productivity positively by using report procedures [Faraj and Sproull, 2000], project management documents, scheduled meetings, and explicit information sharing as part of the development methodology [Sawyer and Guinan, 1998].

Sawyer and Guinan [1998] found that a high level of informal communication among team members and stakeholders, as well as their supportiveness, affect productivity positively. Supportiveness concerns the support that team members give to the other members; the loyalty and help given among team members. Beyond communication, Carey and Kacmar [1997] found that face-to-face communication is more effective than teleconferencing to accomplish complex tasks. Moreover, even with more social transactions, face-to-face communication can be useful in long-term project results. Teasley *et al.* [2000] discuss the importance of team collocation to improve productivity by promoting continuous communication, learning and team building.

Regarding resources constraints, most researchers agree that productivity decreases with increasing constraints. Vosburgh *et al.* [1984] found that productivity decreases in the presence of two or more resource constraints. Maxwell *et al.* [1996] also found that constraints such as timing, reliability, storage, team and project duration impact productivity negatively. Although several other studies [Blackburn *et al.*, 2000, 1996, Brooks, 1995] consider team size as a negative factor because of the high number of necessary communication links, Smith *et al.* [2001] claim that large teams may also be productive, if the developers work independently. Schedule compression [Boehm, 2000, Potok and Vouk, 1998] or expansion [Blackburn *et al.*, 2000, Maxwell *et al.*, 1996] may also affect productivity negatively, as well as staff turnover [Abdel-Hamid, 1989].

Table 3.7: *Project Factors influencing software productivity*

Productivity factors	Positive impact	Negative impact	Neutral
Resource constraints such as timing, reliability, storage, team size, and project duration.		[Maxwell *et al.*, 1996, Vosburgh *et al.*, 1984]	
Schedule compression and expansion.		[Blackburn *et al.*, 2000, Boehm, 2000, Maxwell *et al.*, 1996, Potok and Vouk, 1998]	
Team composition including team size, team collocation, and staff turnover.	[Teasley *et al.*, 2000]	[Abdel-Hamid and Madnick, 1991, Blackburn *et al.*, 2000, 1996, Maxwell *et al.*, 1996]	[Smith *et al.*, 2001]
Communication including informal and face-to-face communication.	[Carey and Kacmar, 1997, Sawyer and Guinan, 1998]		

3.3.4 Process factors

Process factors include software methods, tools, customer participation, software lifecycle, and reuse. Table 3.8 presents the results of our literature research on process factors influencing software productivity.

In this thesis, we consider development methods and life cycle, specific techniques (e.g., documentation, early prototyping), and tools usage as process factors. During the last decades, researchers have argued that software processes impact on productivity by providing guidelines to carry out some tasks and eliminating waste and rework [Boehm, 1987, Maxwell and Forselius, 2000, Sawyer and Guinan, 1998]. However, this is not a consensus [Faraj and Sproull, 2000]. On the one hand, Boehm [1981] suggests iterative and incremental development approaches to increase productivity. On the other hand, Tan *et al.* [2009] hypothesize that productivity tends to decrease along

the iterations, probably due to the increasing difficulty in integrating the parts of the system as it grows.

Some techniques, such as early prototyping [Blackburn *et al.*, 1996, MacCormack *et al.*, 2003, 2001], are more accepted as positive factors influencing productivity. Moreover, daily builds may explain significantly productivity variations, especially when it interacts with other practices, such as early prototyping [MacCormack *et al.*, 2003].

Others practices, such as tools usage, show contradictory results. Some authors did not find a significant relationship between tools (e.g., case tools) and productivity [Banker *et al.*, 1991, Blackburn *et al.*, 2000, Jiang *et al.*, 2007, Sawyer and Guinan, 1998]. Developers' resistance and insufficient training are possible explanations. Other researchers [Bruckhaus *et al.*, 1996, Guinan *et al.*, 1998] believe that the benefits from the tools usage depend on how much effort they generate in the original work process.

Table 3.8: *Process Factors influencing software productivity*

Productivity factors	Positive impact	Negative impact	Neutral
Customer participation refers to real customer involvement in project.	[Vosburgh *et al.*, 1984]	[Maxwell and Forselius, 2000]	
Daily builds, the frequent integration of system components.	[MacCormack *et al.*, 2003]		
Documentation, the use of artifacts to register project and product knowledge.	[Boehm, 2000]	[Jones, 2000]	
Early prototyping involves prototyping efforts to resolve potential high-risk issues.	[Blackburn *et al.*, 1996, MacCormack *et al.*, 2003, 2001]		
Incremental and iterative development , diversity of ways of producing parts of a system, trying them out, and feedback on user experience of production of new or revised parts.		[Tan *et al.*, 2009]	
Modern programming practices such as use of top-down requirements analysis and design structured design notation, structured code etc.	[Boehm, 1981, Vosburgh *et al.*, 1984]		
Programming language abstraction, e.g., Java is a high level language, more abstract than C.			[Blackburn *et al.*, 1996, Maxwell *et al.*, 1996]
Software methods and practices			[Faraj and Sproull, 2000]
Tools usage such as CASE tools, IDEs, etc.	[Boehm, 2000]		[Banker *et al.*, 1991, Blackburn *et al.*, 2000, 1996, Bruckhaus *et al.*, 1996, Guinan *et al.*, 1998, Jiang and Comstock, 2007, Sawyer and Guinan, 1998]

3.4 Agile team productivity

A team can be defined as "a small number of people with complementary skills who are committed to a common purpose, performance goals, and approach for which they hold themselves mutually accountable" [Katzenbach and Smith, 2005]. Teams are supposed to be better suitable for executing complex tasks because team members share workload, observe behavior of other team members and contribute to the sub-tasks of the complex task [Mathieu *et al.*, 2000].

Most organizations use teams in the organizational activities, and software development relies predominantly on teamwork [Tang and Kishore, 2010]. According to Salas *et al.* [2004], teamwork is "a set of interrelated thoughts, actions, and feelings that combine to facilitate coordinated, adaptive performance and the completion of taskwork objectives".

In general, a team evolves during its existence. Team members must learn new behaviors and skills to improve their work and create a high-performance team. Several models describe the team development stage, such as the Tuckman [1965] classic model of forming, storming, norming, and performing. In the forming stage, team members attempt to create social and task structures to guide their interactions. When they realize that it is difficult to create consensus on a certain approach, they shift to a storming stage, in which different members compete for influence. The team evolves and reconciles differences by setting norms to guide their interactions. Once the norms are well established, members can focus on achieving common goals.

Team productivity is a function of what individual team members are doing, and it is more than the sum of all the individual bests of team members [Katzenbach and Smith, 2005, Sudhakar *et al.*, 2011]. Teams only exist to perform goals better than an individual. In this context, the individual productivity is not as important as team productivity. In fact, the evaluation of individual productivity may not affect the productivity of other knowledge workers [Bosch-Sijtsema *et al.*, 2009]. These assumptions provide a motivation to study productivity of *teams'*, not individuals. In addition, evaluating productivity at the team level avoids measurement dysfunctions, since the team goals are the main target, not the individual work.

Finally, we argue that *Knowledge worker productivity* is the best way to describe productivity in the software development context, including agile teams. Nowadays, productivity goes beyond the mere relationship between outputs and inputs. We discussed in Section 2.4, the paradoxical relationship between efficiency and flexibility, leading to productivity. In Section 3.1, we showed that KW productivity encompasses more than traditional productivity, profitability, and performance dimensions, since it includes autonomy, learning, creativity, and innovation. All of them closely related to the Agile Manifesto's principles [Beck *et al.*, 2001].

3.4.1 Monitoring software productivity of self-managed teams

The construct of monitoring has roots in both leadership and self-regulation literature [Kozlowski *et al.*, 2008, Marks and Panzer, 2004]. In a self-managing team, members have responsibility not only for executing task, but also for monitoring, managing, and improving their own performance. Monitoring is, therefore, one of the functions of self-managing teams, having an important function for team regulation and thus for performance, particularly in dynamic work environments [Marks *et al.*, 2001, Marks and Panzer, 2004].

Marks and Panzer [2004] found that team monitoring improves coordination and feedback processes, which in turn improve team performance. Monitoring team performance is considered one of the critical factors for successful implementation of high performance teams, which should be mutually agreed by team and management [Castka *et al.*, 2001].

Monitoring is considered one of team's action processes that is performed in all team developmental phases. According to Kozlowski *et al.* [2008], *team performance monitoring* is a type of monitoring that occurs at the *team development* phase, i.e., a phase after the team formation and task and role development. In the team development phase, the *teamwork* capability is developed and attitudes such as mutual trust and respect are established. Once a team crosses the team development phase, they are ready to develop adaptive capabilities, going into the last developmental

phase: team improvement. Therefore, *teams should be developed to be adaptive*. Monitoring team performance is one of the team actions that can be used to reach adaptive capabilities.

We discuss below monitoring action processes according to the taxonomy of team processes proposed by Marks *et al.* [2001].

Monitoring progress toward action. Tracking task and progress toward mission accomplishment, interpreting system information in terms of what needs to be accomplished for goal attainment, and transmitting progress to team members.

Systems monitoring. Tracking team resources and environmental conditions as they relate to mission accomplishment, which involves (1) internal systems monitoring (tracking team resources such as personnel, equipment, and other information that is generated or contained within the team), and (2) environmental monitoring (tracking the environmental conditions relevant to the team);

Team monitoring and backup behavior. Assisting team members to perform their tasks. Assistance may occur by (1) providing a teammate verbal feedback or coaching, (2) helping a teammate behaviorally in carrying out actions, or (3) assuming and completing a task for a teammate.

Team monitoring is primarily a cognitive operation in which team members observe actions of their teammates and watch for errors or performance discrepancies [Marks and Panzer, 2004]. Team members monitor their teammates' execution of critically interdependent roles to reach team goals and/or sustain acceptable levels of team performance [Dickinson and McIntyre, 1997, Salas and Cannon-Bowers, 1996].

The following benefits of monitoring are then advocated in the context of teamwork:

- Members who monitor effectively should be better capable of obtaining team situation awareness and to analyze rhythm, timing, and pace of team member activities, which facilitates coordination [Marks and Panzer, 2004].
- Enables individuals to recognize when their team members make mistakes or perform inadequately, which should promote assistance to teammates when needed [Marks and Panzer, 2004, Yeatts and Hyten, 1998].
- Team members appeared to take ownership of any performance problems and to take on responsibility for improving areas that were not up to performance goals [Yeatts and Hyten, 1998]. This appeared to result in better focus of team efforts, talents, resources, and procedures because teams were routinely evaluating whether its current work processes were resulting in high performance [Yeatts and Hyten, 1998].

Finally, monitoring measures of team performance was found to be extremely important to team performance [Castka *et al.*, 2001, Yeatts and Hyten, 1998]. High performance teams establish performance goals through *measurable* criteria and monitor their success in achieving their performance goals during frequently team meetings. Thus, it is fundamental to understand software productivity measurement to put the productivity monitoring in practice.

Proposition 4: Monitoring team performance is a team action to reach adaptive capabilities. Measuring is part of a continuous productivity monitoring for self-managed adaptive teams.

Related work in agile team productivity monitoring

Despite monitoring being recognized as an important process for teamwork, little is known about team monitoring and agile team productivity. We found one study on agile team monitoring conducted by Moe *et al.* [2010]. They recognize that Scrum is not very specific on how to establish monitoring in development teams, although this is implicitly a prerequisite for feedback, coordination, and backup behaviour in self-organized teams.

Agile methods rely on teamwork to face turbulences and to develop adaptive capabilities. Since teamwork performance (and productivity) in dynamic environments depends on monitoring processes, there is a need for more studies on agile team monitoring.

Proposition 5: Agile productivity monitoring processes are important action processes to develop teams towards adaptive capabilities - a fundamental attribute for agile teams.

3.4.2 Agile team productivity measurement

Although many productivity metrics are available (see Section 3.2.2), new metrics to approach measurement in an agile environment were proposed by agile champions, such as Kent Beck and Martin Fowler. There is anecdotal evidence that these metrics are adopted in the industry [VersionOne, 2010]. Most of these metrics were proposed as part of the *tracking* activity, a practice recommended by some agile methods (e.g., XP [Beck and Andres, 2004] and Crystal Clear [Cockburn, 2006]).

Table 3.9 summarizes some agile productivity metrics, classifying them according to our definition of productivity (Section 3.1) and agility (Section 2.3). It is important to note that the agility concept brings out lean principles and related metrics. Thus, we consider both agile and lean metrics as possible productivity metrics for agile teams. Our objective here is not to present an exhaustive list of agile productivity metrics, but just to describe some agile measurement initiatives.

Table 3.9: *Agile metrics and applications in productivity evaluation*

Dimension	Metric	Key reference
Quality	Defect count	[Junior and Meira, 2009]
	Technical Debt	[Gat, 2010].
	Faults-Slip-Through	[Damm *et al.*, 2006, Petersen, 2011]
Value	Customer Satisfaction Survey	[Behrens, 2009]
	Business Value Delivered	[Cohn, 2005]
Leanness	Lead time	[Poppendieck and Poppendieck, 2006]
	Queues	[Poppendieck and Poppendieck, 2006]
	Work in Progress (WIP)	[Behrens, 2009]
	Flow	[Poppendieck and Poppendieck, 2006]
Efficiency	Waste	[Poppendieck and Poppendieck, 2003]
Speed	Velocity	[Beck and Andres, 2004]
	Acceleration	[Ambler, 2008].

Quality. *Defect count* may be collected in various stages of development. Counting defects in individual iterations can produce a fairly large variation and in the collected numbers may paint a misleading picture [Junior and Meira, 2009]. *Technical Debt* is a metaphor referring to

the consequences of taking shortcuts in the software development, for example, code written in haste that is in need of refactoring. The debt can be represented in financial figures, which makes the metric suitable to communicate to upper management [Gat, 2010]. *Faults-Slip-Through* measures the test efficiency by where the defects should have been found and where it actually was. It monitors how well the test process works and addresses the cost savings of finding defects early. In case studies on implementation of lean metrics, the faults-slip-through has been recommended as the quality metric of choice [Damm *et al.*, 2006, Petersen, 2011].

Value. *Customer Satisfaction Survey* is asking the customer if the features are actually useful. It has proved useful to survey the customer over the time of a release and is much in line with the agile principle of customer cooperation [Behrens, 2009]. *Business Value Delivered* includes estimations and is not an exact science. A possible solution to measure the business value involves dividing the business case's value between the tasks. The delivery of value can be displayed in a Business Value Burnup Chart [Cohn, 2005].

Leanness. *Lead time* is the amount of time that elapses between when a process starts and when it is completed. It relates to the financial metric throughput. The lead time should be as short and stable as possible. It reduces the risk that the requirements become outdated and provides predictability. The metric is supported by Poppendieck and Poppendieck [2006], who state that the most important to measure is the "concept-to-cash-time" together with financial metrics. In software development, *queue* time is a large part of the lead time. In contrast to the lead time, queue metrics are leading indicators. Large queues indicate that the future lead time will be long, which enables preventive actions. By calculating the cost of delay of the items in the queues, precedence can be given to the most urgent ones [Poppendieck and Poppendieck, 2006]. *Work in Progress* (WIP) is one of the best ways to monitor large queues. If used in combination with queue metrics, WIP constraints prevent dysfunctional behavior such as simply renaming the objects in queues to work in progress. The metric is also an indicator of how well the team collaborates. A low WIP shows that the team works together on the same tasks [Behrens, 2009]. Finally, improving the *flow* means shorter lead-times and thus timely delivery of value to the customer. In manufacturing, cumulative flow diagrams have been proposed to visualize the flow of material through the process [Petersen and Wohlin, 2011].

Efficiency. Eliminating *waste* is one way to increase productivity. There are seven types of waste in software development [Poppendieck and Poppendieck, 2003]: i) Partially Done Work, ii) Extra Features, iii) Relearning, iv) Handoffs, v) Delays, vi) Task Switching, vii) Defects. Similar metrics were widely adopted in lean manufacturing systems. By revealing waste in projects, significant actionable opportunities in terms of saving resources and accelerating lead-time can be reached for practical use [Ikonen *et al.*, 2010].

Speed. In several of the agile methods, the *velocity* of delivered requirements is used to achieve predictability and estimate the delivery capacity [Beck and Andres, 2004]. Acceleration is one measurement for the relative change in the team velocity metric, indicating a change in productivity [Ambler, 2008].

Nevertheless, there is little empirical evidence supporting these metrics in the agile software development environment. Most evidence is either still anecdotal, based on agile champions from industry, or poorly examined by academic research. Thus, we do not know how those metrics really support agile team productivity monitoring and measurement process. As we discussed before, productivity measurement and monitoring should be used to help teams to learn and improve. Also, there is no evidence of how these metrics would be chosen considering agility, i.e., the need for team adaptation. Depending on the context, teams will face different challenges and goals, which impacts on how they will evaluate their own productivity.

Related work in agile team productivity measurement

In a systematic review on agile methods empirical research conducted by Dybå and Dingsoyr [2008], only studies that analyze team productivity in terms of lines of code (LOC) per time unit were reported. Measuring productivity with a richer set of metrics would allow better and deeper analysis of the results obtained in the studies. Additionally, doing so would contribute to the selection of productivity metrics best fit to evaluate agile methods.

Miranda and Bourque [2010] state that there is a need to collect, measure, and present progress information in software development projects, and Agile projects are no exception. They explore an indicator called "line of balance" (LOB) to gain insights into the progress of projects not provided by burn down charts or cumulative flow diagrams (two of the most common indicators used to track and report progress in Agile projects). The indicator aims to tell more about what is going on inside the project. According to Miranda and Bourque [2010], the advantages of the LOB method are: it shows not only what has been achieved, but also what was supposed to be achieved in a single chart; ii) shows work in progress, permitting a more accurate assessment of the project status; and iii) exposes process bottlenecks, allowing the team and the project sponsor to focus on the points causing slippages. Thus, the indicator would be used to evaluate *leanness*, one dimension of agility.

Staron and Meding [2011a] studied how one can identify and monitor bottlenecks in software development projects in order to prevent inefficiency. Bottlenecks are often related to *leanness*. They found that measures used to monitor the flow do not need to be very complex and, by changing the perception of well-known phenomena (e.g., using throughput instead of productivity), the development of the measurement system is straightforward and is not prone to generate conflicts. They argue that it was much harder to achieve consensus on how the productivity should be measured and how it should be compared. However, as discussed previously, there is a movement to measure productivity for learning and improving, not for controlling and comparing. They also argue that capacity should be monitored, not speed. However, speed is one of dimensions of agility. Again they explain that the consensus was harder to obtain. Maybe the choice of productivity metrics is critical to obtain in such teams, although there is little support for this assumption in the literature.

Ktata and Lévesque [2010] describe an approach used in an agile software engineering company to design and initiate a measurement program taking into account the specificities of their agile environment, principles and values. They found that agile teams have different needs when trying to establish a measurement program in comparison to traditional teams. Agile teams and other work groups are reluctant and resistant to change. Finally, agile people are more interested in value delivery, technical debt, and multiple aspects related to team dynamics and will cooperate to the collection of data only if can be done automatically. Thus, *quality* and *value* were the primary dimensions of measurement. Ktata and Lévesque [2010] highlight three risks of failure in implementing measurement programs in agile environments: resistance to change, a fear of having increased overhead activities to support data collection, and choosing the inappropriate set of indicators.

In contrast, Petersen and Wohlin [2011] showed that practitioners found flow measures useful and identify improvements based on them, which drove the organization toward the use of lean and agile principles. As we mentioned, flow is related to *leanness*. The participants found the measures useful in seeing the progress of development for complex products where many tasks are executed in parallel. They incorporated the measures as part of the improvement work at the studied company. Metrics were specifically useful on: (1) requirements prioritization; (2) the measures aid in allocating staff; (3) the measures provide transparency for teams and project managers of what work is to be done in the future and what has been completed; and (4) software process improvement drivers can use the measures as indicators to identify problems and achieve improvements from a long-term perspective.

Regarding performance measurement systems, Highsmith [2006] proposed the concept of *Adaptive performance management*. It comprises an adaptive performance management system divided into two parts: (1) the *project* performance management system, which focuses on outcomes that

generate customer value; and (2) the *team* performance management system, which focuses on informational metrics that teams can use to improve. He refers to the Austin's work [Austin, 1996] on performance measurement, outlining the importance of avoiding measurement dysfunctions (we discussed dysfunctions also in Section 3.2.1). It is very important to note the separation between project performance and team performance. They should be measured and managed in different ways.

Highsmith [2006] also refers to Beyond Budgeting [Hope and Fraser, 2003], a performance management model from the accounting literature. This model is conceptually similar and appears to align well with agile methods, since it aims to deal with a changing business and operating environment [Lohan *et al.*, 2010]. The model consists of leadership and process principles that should be used together in a holistic way, improving organizational performance management [Bogsnes, 2008]. Beyond Budgeting advocates a high-performance climate based on relative success, accounting them for customer outcomes; people freedom to make local decisions that are consistent with governance principles and the organization's goals; place the responsibility for value-creating decisions on frontline teams; and open and ethical information systems.

Lohan *et al.* [2010] investigated the organizational issues and challenges that affect the performance of agile software development teams through the lens of Beyond Budgeting. They found that traditional organizational structures and processes intended to improve performance might end up impeding the performance of the agile team. They call for research on ways in which traditional command-and-control organizations can adapt their practices to achieve a more agile friendly environment.

Finally, Highsmith [2006] states: "it is unrealistic to assume that agility at lower levels in the organization can be fully implemented without a change, at the very top of an enterprise, in how performance is measured". However, little is known about how to implement adaptive performance management at the team level. We aim to investigate how to monitor productivity factors, considering agility and adaptability, and also investigate how agile metrics support this monitoring (RQ3). Our results will surely contribute to the adaptive performance management issue.

3.4.3 Agile productivity factors

Petersen [2011] identified a need for studies that provide good knowledge about productivity factors. In particular, new factors have to be considered with the change in developing software, and old factors need to be re-evaluated in the new context. Thus, new productivity factors may appear or old ones may become less important as the context changes. Factors should be continuous evaluated through *objective* or *subjective* measurement [Pries-Heje and Commisso, 2010, Sudhakar *et al.*, 2011].

Therefore, an important contribution would be developing a process to monitor productivity factors. Considering the agile software development environment, this monitoring process must take into account agility variables, such as flexibility, learning, and responsiveness (as discussed in Chapter 2, Section 2.3).

> *Proposition 6:* Developing a process to monitor productivity factors must take into account agility variables, such as flexibility, learning, and responsiveness.

Another important issue is using a representative conceptual framework to study productivity factors in agile development environments. Understanding the nature of agile teams, such as self-organization, teamwork, and autonomy may help to design an adequate framework. Since our Research Question 2 aims to explore agile team productivity factors, we must develop such framework based on appropriated literature.

Proposition 7: Agile team productivity factors should be investigated in light of a representative conceptual framework based on the nature of agile teams, such as self-organization, teamwork, and autonomy.

3.4.4 Summary of propositions regarding agile team productivity

We presented a review on productivity definitions, measurement, factors, and team productivity in dynamic environments. We described propositions from the review rationale, all of them outlining important issues for team productivity evaluation in knowledge-workers self-managed settings. Then we discussed how these concepts are aligned to agile teams. The propositions are summarized bellow:

Proposition 1: Productivity measurement should be used predominantly for learning, short-term management, and long-term improvement. As it is a dynamic measure by nature, it may differ with the products and the technology used. The set of metrics may vary at different stages of a product life cycle and should be developed in a process-oriented, bottom-up approach.

Proposition 2: Software development productivity should be evaluated in a holistic way, since it involves different perspectives, human knowledge, learning, creativity, as well as performance and profitability.

Proposition 3: It is possible to develop a Performance Measurement System within agile teams without generating additional overhead, since metrics can be chosen, integrated, and analyzed incrementally, as it is normally done in an agile project.

Proposition 4: Monitoring team performance is a team action to reach adaptive capabilities. Measuring is part of a continuous productivity monitoring for self-managed adaptive teams.

Proposition 5: Agile productivity monitoring processes are important action processes to develop teams towards adaptive capabilities - a fundamental attribute for agile teams.

Proposition 6: Developing a process to monitor productivity factors must take into account agility variables, such as flexibility, learning, and responsiveness.

Proposition 7: Agile team productivity factors should be investigated in light of a representative conceptual framework based on the nature of agile teams, such as self-organization, teamwork, and autonomy.

These propositions will guide our research by supporting the development of a conceptual framework to focus the research design, particularly useful to focus the data collection, facilitating data analysis.

3.5 Summary and the Challenges of this Thesis

This chapter has shown software productivity definitions, also adopting knowledge worker productivity as a formal definition of agile team productivity. We presented the most important aspects related to productivity, which are: definitions, measurement, monitoring, and factors. We also discussed these aspects in the agile context, discussing productivity of teams in dynamic environments, their internal processes, such as monitoring, and finally, some popular metrics.

Some challenges in agile software productivity were discussed in the previous sections. This section describes which of these challenges are the subjects of this thesis in the context of agile team productivity evaluation. The Research Challenges (RCs) can be defined as:

RC1. Productivity is hard to measure. Productivity of knowledge workers is hard to measure because it involves many human aspects, such as creativity, autonomy, and innovation. However, productivity measurement may help team learning and improving. Thus, we need to figure out how to choose a good set of objective and subjective measurement that enable learning and improvement.

RC2. There are many possible productivity factors. Since there are many possible productivity factors impacting an agile team, we need to focus our data collection and analysis not to get lost during the research process. Two strategies should be developed to cope with this issue: i) creating a conceptual framework to guide the data collection and analysis, and ii) observing multiple teams to enable the emergence of more influential productivity factors.

RC3. It is challenging to develop adaptive productivity measurement systems. Our goal is not to develop a new productivity measurement system for organizations, but only for agile *teams*. However, to answer RQ3 ("How to monitor productivity factors, considering agility and adaptability? How do agile metrics support productivity monitoring?"), we probably will follow the same steps of a PMS deployment and evaluation (Sections 3.2.3 and 3.4.2). There are also few previous studies on this topic, which is *per se* a challenge.

RC4. Developing a productivity measurement system for the IT industry. Another important issue is that we aim to develop the productivity measurement system in partnership with the IT industry. This would create additional risks to the research, since any partnership is vulnerable to changes in both sides. We also have to learn as much as we can about the organizations to succeed with the project.

RC5. Validating the productivity measurement system. Since the measurement system aims to promote learning and improvement (instead of controlling) in agile teams, we need to develop ways for identifying the impact of metrics on these two dimensions. Both are not a 'quick and easy' outcomes to validate.

Chapter 4

Research method and design

This chapter provides an introduction to research approaches and strategies. It also discusses advantages and challenges in the approaches adopted in this thesis: case studies, survey, and action research. We discuss also validity threats for all selected scientific approaches. Finally, the challenges are discussed, facing empirical studies, in general, and in this thesis, in particular, in selecting research methods.

4.1 Research Strategies in Empirical Research

Empirical science concerns the acquisition of knowledge by empirical methods [Sjøberg *et al.*, 2007]. Evidence-based software engineering (EBSE) aims to provide the means by which current best evidence from research can be integrated with practical experience and human values in the decision making process regarding the development and maintenance of software [Kitchenham, 2004]. According to Neuman [2006], there can be three types of purposes for a study: exploratory, descriptive, and explanatory. Table 4.1 summarizes the main characteristics of each purpose.

Table 4.1: *Purpose of research [Neuman, 2006]*

Exploratory	Descriptive	Explanatory
• Become familiar with the basic facts, setting, and concerns. • Create a general mental picture of conditions. • Formulate and focus questions for future research. • Generate new ideas, conjectures, or hypotheses. • Determine the feasibility of conducting research. • Develop techniques for measuring and locating future data.	• Provide a detailed, highly accurate picture. • Locate new data that contradict past data. • Create a set of categories or classify types. • Clarify a sequence of steps or stages. • Document a causal process or mechanism. • Report on the background or context of a situation	• Test a theory's predictions or principle. • Elaborate and enrich a theory's explanation. • Extend a theory to new issues or topics. • Support or refute an explanation or prediction. • Link issue of topics with a general principle. • Determine which of several explanation is best

There are three types of research paradigms that have different approaches to empirical studies [Creswell, 2003, Seaman, 1999, Wohlin *et al.*, 2000]:

- *Qualitative research* is concerned with studying objects in their natural setting. A qualitative researcher attempts to interpret a phenomenon based on explanations that people bring to

them.

- *Quantitative research* is concerned with discovering causes noticed by the subject in the study, and understanding their view of the problem at hand. A quantitative study is mainly concerned with quantifying a relationship or with comparing two or more groups. Quantitative research is often conducted through setting up controlled experiments or collecting data through case studies or surveys.

- The *mixed-method approach* is evolved to compensate for limitations and biases of the above mentioned strategies, seeking convergence across other methods. The combination of quantitative and qualitative methods is usually more fruitful than either one in isolation. How to combine the qualitative and quantitative methods in the design is discussed by Seaman [1999].

Considering our research questions enunciated in Section 1.2, we have a mix of question types. RQ1 is both exploratory and explanatory. We designed a *multiple-case study* for the exploratory subquestion (*how do companies define productivity?*), since we argue in the previous chapters that traditional definitions of productivity do not fit well into the agile context. For the other subquestion (*How important is productivity for companies adopting agile methods*), we chose the *survey* method approach, because the question is about *how much*.

RQ2 is essentially exploratory (*what factors impact agile team productivity and how is this impact from the team point of view? Which agile practices are perceived to impact on a given team's productivity?*, so we included it in the multiple-case studies. Finally, RQ3 is exploratory, but also involves deploying new methods in a real company and learning throughout the process (*How to monitor productivity factors, considering agility and adaptability? How do agile metrics support productivity monitoring?*); for this, we chose the *action research* approach. We detail these three approaches in the next subsections, as well as the research design for each research phase.

4.1.1 The survey approach

The survey research method comprises many details; there is an extensive literature covering the topic (e.g., [Dillman, 2006, Fink, 2003, Groves, 1989, Selm and Jankowski, 2006]). In this section we will give a short overview of the method, highlighting the main issues, advantages, and possible threats.

A survey is not just the instrument (the questionnaire or checklist) for gathering information [Kitchenham and Pfleeger, 2008]. It is a system for collecting information from or about people to describe, compare, or explain their knowledge, attitudes, and behavior [Fink, 2003]. The term 'survey' is used in a variety of ways, but generally refers to the selection of a relatively large sample of people from a pre-determined population (group of people in whom the researcher is interested in a particular study), followed by the collection of a relatively small amount of data from those individuals [Kelley *et al.*, 2003].

In contrast to other research methods, such as case studies or experiments, survey research is most appropriate when [Pinsonneault and Kraemer, 1993]:

- the central questions of interest about the phenomena are "*what is happening?*", and "*how and why is it happening?*". Survey research is especially well-suited for answering questions about *what*, *how much*, and *how many* and, to a greater extent than is commonly understood, questions about *how* and *why*;

- Control of the independent and dependent variables is not possible or not desirable;

- The phenomena of interest must be studied in its *natural setting*;

- The phenomena of interest occur in *current time* or the *recent past*.

The main steps when conducting a survey are [Kitchenham and Pfleeger, 2008]:

- Setting the goals;
- Survey design;
- Developing the survey instrument (i.e., the questionnaire);
- Evaluating the survey instrument;
- Obtaining valid data;
- Analysing the data.

According to Fink [2003], survey goals can come from reviews of literature and other surveys. A systematic review of the literature will tell what is the current knowledge in the field. Using the available data, the researcher can figure out where are the gaps which need to be filled, which could be the survey goal. Goals can also come from experts, individuals who are knowledgeable in the field and that can be affected by its outcomes or are influential in implementing its findings. In general, those people are interviewed in focus groups to identify objectives and further research hypotheses. There are three common types of objectives [Kitchenham and Pfleeger, 2008] as follows:

- To *evaluate the rate or frequency of some characteristic* that occurs in a population. For example, we might be interested in the frequency of failing projects.
- To *assess the severity of some characteristic or condition* that occurs in a population. For example, we might be interested in the average overrun of software projects.
- To *identify factors that influence a characteristic or condition.* For example, we might be interested in factors that predispose a process improvement activity towards failure or towards success.

As Kitchenham and Pfleeger [2008] explains, the first two types of survey objective are descriptive, since they describe some condition or factor found in a population in terms of its frequency and impact. The last type of survey looks at the relationship existing among factors and conditions within a population.

Regarding the survey design, there are two common types of design [Kitchenham and Pfleeger, 2008]:

- *Cross sectional*: In this type of study, participants are asked for information at one fixed point in time. For instance, we may poll all the members of a software development organization at 10-am on a particular Monday, to find out what activities they are working with on that morning. This information gives us a snapshot of what is going on in the organization.
- *Longitudinal*: This type of study is forward-looking, providing information about changes in a specific population over time. There are two main variants of longitudinal designs, one can survey the same people at each time period or survey different people.

The other issue to decide is the way in which the survey will be administered. The traditional methods include telephone interviewing, self-administered mail questionnaires, and face-to-face interviewing. New techniques were matured with the development of the Internet, most notably e-mail and Web-based surveys [Dillman, 2006, Selm and Jankowski, 2006].

Each of these survey administration methods has its own advantages and disadvantages. Telephone surveys, for instance, allow the querying of complex questions and cover a wide geographic area. But it often is confused with sales calls, and is considered intrusive. In this thesis, we are more interested in online surveys, because they are an effective way to gather information quickly and relatively inexpensively from a large geographic region.

After these choices, the survey instrument should be developed. Survey instruments are usually questionnaires and they are developed using the following steps: i) Search the relevant literature;

ii) Construct (create or re-use) an instrument; ii) Evaluate the instrument, and iv) Document the instrument. Kitchenham and Pfleeger [2008] provide useful explanations for each step.

Once the instrument is created, it is essential that we evaluate it [Kitchenham and Pfleeger, 2008]. Evaluation is often called pre-testing, and it has several different goals. First, to check if the questions are understandable. Also, we need to assess the likely response rate and the effectiveness of the follow-up procedures. Then, it is necessary to evaluate the reliability and validity of the instrument. A survey is reliable if we administer it many times and get roughly the same distribution of results each time [Kitchenham and Pfleeger, 2008]. Validity refers to the degree to which a study accurately reflects or assesses the specific concept that the researcher is attempting to measure. It is an important measure of a survey instrument's accuracy (usually assessed as face validity, content validity, criterion validity, and construct validity) [Fink, 2003]. A measure may be reliable (because it gets always the same distribution) but not valid (because it does not reflect the intended constructs); but it cannot be valid without being reliable. That is, reliability is necessary, but not a sufficient condition for validity. Finally, we need to ensure that our data analysis techniques match our expected responses. In general, survey instruments are evaluated through pilot studies (administration of the survey to a smaller sample) or focus groups (mediated discussion groups).

Then, the survey is administered. Some authors give specific advice to this phase [Dillman, 2006, Fink, 2003], such as how to promote the participation and how to thank those who have participated. After the period of data collection, the survey should be analyzed. Kitchenham and Pfleeger [2008] states: "if you have designed your survey properly, you should have already identified the main analysis procedures". There are many types of survey analysis. We can partition the responses into more homogeneous groups to enable comparisons and specific reports by groups. We also can separate the analysis by the data type. Some data analysis are easier, such as numerical ones. But analyzing ordinal and nominal data can be tricky, due to their intrinsic mathematical properties.

Advantages and disadvantages

The main advantages of surveys as a research methods are [Kelley *et al.*, 2003]:

- the research produces data from real-world observations (empirical data);
- the breadth of coverage of many people or events means it is more likely than some other approaches to obtain data based on a representative sample, and can, therefore, be generalized to a population;
- surveys can produce a large amount of data in a short time for a fairly low cost. Researchers can set a finite time-span for a project, which can assist in planning and delivering end results.

Although surveys are an extremely common research method, survey based research is not an easy option [Kitchenham and Pfleeger, 2008]. Some disadvantages are:

- A methodology relying on standardization forces the researcher to develop questions general enough to be minimally appropriate for all respondents, possibly missing what is most appropriate to many respondents. Surveys are inflexible in that they require the initial study design (the tool and administration of the tool) to remain unchanged throughout the data collection.
- The researcher must ensure that a large number of the selected sample will reply.
- It may be hard for participants to recall information or to tell the truth about a controversial question.

Threats to the validity of surveys

Four sources of survey errors threat the validity of survey results [Dillman, 2006]:

Coverage Error. The result of not allowing all members of the survey population to have an equal or known chance of being sampled for participation in the survey. There are three types of population [Groves, 1989]: i) the *population of inference* is the population on which the researcher ultimately intends to draw conclusions; ii) the *target population* is the population of inference less various groups that the researcher has chosen to disregard; iii) the *frame population* is a portion of the target population that can be enumerated prior to the selection of the sample. The survey sample then consists of those members of the sampling frame that were chosen to be surveyed and coverage error is the difference between the frame population and the population of inference.

Sampling Error. The result of surveying only some, and not all, (randomly selected) elements of the survey population. It arises when a subset of the target population is surveyed yet inference is made about the entire population. Assuming no difference between the population of inference and the target population, the sampling error is simply a quantification of the uncertainty in the sample statistic.

Measurement Error. The result of poor question wording or questions being presented in such a way that inaccurate or uninterpretable answers are obtained. These arise from [Groves, 1989]: (a) effects of interviewers on the respondents' answers to survey questions; (b) error due to respondents, from the inability to answer questions, lack of requisite effort to obtain the correct answer, or other psychological factors; (c) error due to the weakness in the wording of survey questionnaires; and, (d) error due to effects of the mode of data collection, the use of face to face or telephone communications'.

Nonresponse Error. The result of people who respond to a survey being different from sampled individuals who did not respond, in a way relevant to the study. Groves [1989] calls nonresponse 'an error of nonobservation'. The response rate, which is the ratio of the number of survey respondents to the number sampled, is often taken as a measure of goodness.

In online surveys, however, the sampling error cannot be calculated since the underlying assumption behind this calculation requires knowledge about the probabilities of selection [Selm and Jankowski, 2006]. Since sampling and coverage errors are difficult to overcome in online surveys, researchers should, according to Dillman [2006], direct effort at decreasing the occurrence of measurement and non-response errors through, for instance, designing respondent-friendly questionnaires [Selm and Jankowski, 2006].

4.1.2 The case study approach

Case connotes a spatially delimited phenomenon (a unit) observed at a single point in time or over some period of time. It comprises the type of phenomenon that an inference attempts to explain. Each case may provide a single observation or multiple (within-case) observations [Gerring, 2006].

Case studies are particularly useful for answering research questions about *how* and *why*, which are more *explanatory* [Yin, 2008]. This is because such questions deal with operational links needing to be traced over time, rather than mere frequencies or incidence. But there are also *exploratory* case studies and *descriptive* case studies.

Case studies are suitable for studying contemporary set of events, over which the investigator has little or no control [Yin, 2008]. For this reason, case studies adopt a *naturalistic inquiry*. It assumes multiple realities; meanings and interpretations; generation of knowledge resulting from the interaction between the inquirer and the respondents (as opposed to the process of generalization proposed by the positivists); patterns of plausible influences inferred from social and behavioral studies; and, finally, it recognizes the influence of value systems in the identification of problems, selection of samples etc., which is clearly a researcher's bias that cannot be ignored [Lincoln and Guba, 1985].

According to Gerring [2006], a *case study* is the intensive study of a single case where the purpose of that study is to shed light on a larger class of cases, i.e., a population. Case study research may incorporate several cases, i.e., *multiple-case studies*. When the emphasis of a study shifts from the individual case to a sample of cases, then we call it as a *cross-case* study. In pursuit of guidelines for each possible perspective in a case study, Cunningham [1997] proposed a classification (as shown in Table 4.2).

Table 4.2: *Different types of case studies [Cunningham, 1997]*

	Intensive cases	**Comparative cases**	**Action Research**
Purpose	To develop theory from intensive exploration	To develop concepts based on case comparisons	To develop concepts that help facilitate the process of change
Assumption	Creativity through comparison with existing theories	Comparison of cases leads to more useful theory	Theory emerges in the process of changing
Situation	Usually evolves out of a researcher's intensive experience with culture or organization	Usually concepts are developed from one case compared with another case	Developing theory to assist practices and future social science
Types	Narratives; Tabulation; Explanatory; Interpretative	Case comparisons; Case survey; Interpretative comparisons	Diagnostic Action Research; Experimental Action Research.

Intensive case studies describe a very intensive understanding of the events and practices of organizations, groups, or even one person. The main goal is to provide a history, description, or interpretation of unique or typical experiences or events. They offer vivid and powerful explanations of events and illustrate examples that are not well captured by existing theories. However, they do not allow comparisons of cases. *Comparative case studies* emphasize the use of contrasting observations from varied settings and highlight the development of clear concepts. This is because there is an assumption that more cases allow replication and extension among individual cases. Finally, in Cunningham's classification, a case study can be an *action research*, which focuses on research and learning through intervening and observing the process of change.

The most realistic research setting is found in action research studies, because the setting of the study is the same as the setting in which the results will be applied for a given organization, apart from the presence of the researchers [Sjøberg *et al.*, 2007]. The setting of industry-based case studies is also very similar to the setting of application, which is a similarity between action research and case studies. The difference is that in the latter, researchers may study phenomena that might not be regarded as very relevant by the companies/settings under study [Sjøberg *et al.*, 2007].

Case studies may be adopted in different philosophical stances, i.e., from different theoretical perspectives. The stance adopted affects which methods the researcher believes lead to acceptable evidence in response to the research question [Easterbrook *et al.*, 2008]. Constructivism, also known as interpretivism, is most closely associated with ethnographies, although constructivists often use exploratory case studies and survey research, too. Critical theorists often use case studies to draw attention to things that need changing. However it is action research that most closely reflects the philosophy of critical theorists. Pragmatists use any available methods and strongly prefer mixed methods research, where several methods are used to shed light on the issue under study. As we mentioned in Chapter 1, the dominant philosophical positions taken in this thesis are *interpretivism* and *pragmatism*. We use mixed method research based on case studies, action research, and survey.

Case study design

There are four general types of case study designs: i) single-case holistic design; ii) single-case embedded design; iii) multiple-case holistic design, and iv) multiple-case embedded design. We already described single and multiple cases in the previous section. The difference is the unit of analysis. If the study (or studies) is analyzing just one type of unit, it is said to be holistic. If it is analyzing multiple units, the design is embedded [Yin, 2008].

Case studies are not tightly dependent on the sampling strategy [Yin, 2008]. A purposive sampling will give better answers to research question in small case studies. In general, the selection of a sampling strategy should be based on the research questions and a conceptual framework.

A very important principle of data collection in case studies is the use of multiple sources of evidence. Te major strength of case study data collection is the opportunity to use many data sources [Yin, 2008]. In fact, the need of using multiple sources in case studies is far more important than in other research approaches. Only this way, the researcher can address a wide range of historical and behavioral issues, one of the most important features of case studies.

Using multiple sources of evidence leads to the development of strategies to converge the different lines of inquiry [Yin, 2008]. This process is known as *triangulation*. Patton [1999] argues that "the logic of triangulation is based on the premise that no single method ever adequately solves the problem of rival explanations. Because each method reveals different aspects of empirical reality, multiple methods of data collection and analysis provide more grist for the research mill". Figure 4.1 illustrates two situations [Yin, 2008]: i) when the researcher really achieved triangulation, and ii) when, despite having many data sources, those address different facts. Triangulation is ideal, but it can also be very expensive [Patton, 1999].

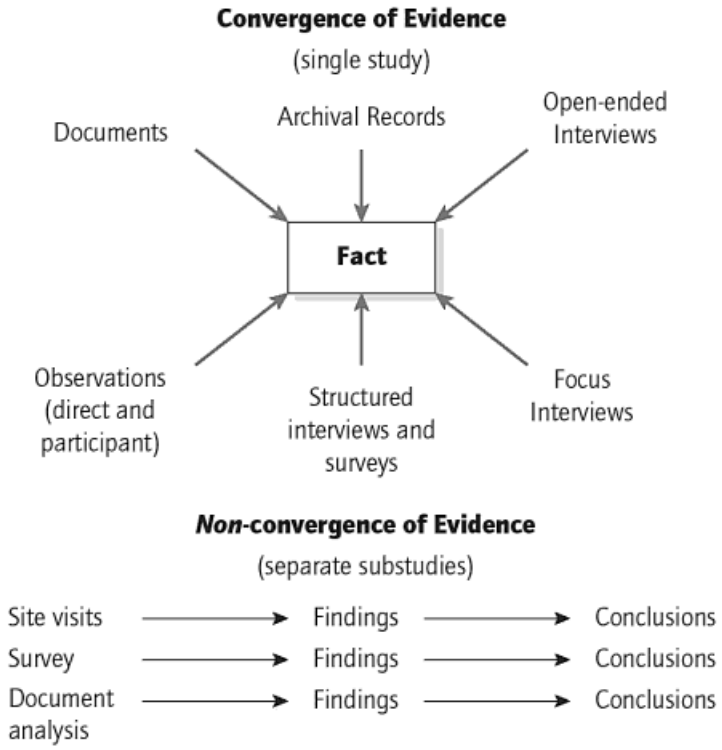

Figure 4.1: *Convergence and Nonconvergence of multiple sources of evidence [Yin, 2008]*

Patton [1999] describes four types of triangulation: i) *methods triangulation* - checking out the consistency of findings generated by different data collection methods; ii) *triangulation of sources* - examining the consistency of different data sources within the same method; iii) *analyst triangulation* - using multiple analysts to review findings; and iv) *theory triangulation* - using multiple perspectives

or theories to interpret the data. Patton [1999] also explains that a common misunderstanding about triangulation is that the point is to demonstrate that different data sources will lead to the same result. Different kinds of data may yield somewhat different results, because "different types of inquiry are sensitive to different real world nuances". Thus, the researcher should shed light in the inconsistencies between data sources, that can make room for deeper insights into the phenomenon under study.

Threats to the validity of case studies

There are four types of threats to validity of case studies, according to Yin [2008]:

Construct validity concerns establishing correct operational measures for the concepts being studied. Three tactics are available to increase construct validity when doing case studies. The first is the use of multiple sources of evidence, in a manner that encourages convergent lines of inquiry, and this tactic is relevant during data collection (triangulation). A second tactic is to establish a chain of evidence, also relevant during data collection. The third tactic is to have the draft case study report reviewed by key informants.

Internal validity concerns establishing a causal relationship, whereby certain conditions are believed to lead to other conditions, as distinguished from spurious relationships. Some tactics are proposed to overcome this threat: the analytic tactic of pattern matching, explanation building, addressing rival explanations, and using logic models.

External validity concerns defining the domain to which a study's findings can be generalized. The external validity problem has been a major barrier when doing case studies. Critics typically state that single cases offer a poor basis for generalization. However, such critics are implicitly contrasting the situation to survey research, in which a sample is intended to generalize to a larger universe. This analogy to samples and universes is incorrect when dealing with case studies. Survey research relies on statistical generalization, whereas case studies rely on *analytic* generalization. However, the generalization is not automatic. A theory must be tested by replicating the findings in a second or even a third setting, where the theory has specified that the same results should occur. Once such direct replications have been made, the results might be accepted as providing strong support for the theory. Researchers often work with multiple-case designs when trying to increase external validity.

Reliability concerns demonstrating that the operations of a study – such as the data collection procedures – can be repeated, with the same results. One prerequisite for allowing this other investigator to repeat an earlier case study is the need to document the procedures followed in the earlier case. The general way of approaching the reliability problem is to make as many steps as operational as possible and to conduct research as if someone were always looking over your shoulder.

4.1.3 The action research approach

There are many definitions of Action Research. One of the most widely cited is that of Rapoport [1970] (p. 499), who defines Action Research in the following way:

> Action research aims to contribute both to the practical concerns of people in an immediate problematic situation and to the goals of social science by joint collaboration within a mutually acceptable ethical framework.

A seminal contribution to the Action Research literature was made by Susman and Evered [1978], with a formal cyclical process model [Davison *et al.*, 2004]. Action research aims to solve current practical problems while expanding scientific knowledge. Unlike other research methods, where the researcher seeks to study organizational phenomena but not to change them, the action

researcher is concerned with creating organizational change and, simultaneously, with studying the process [Baburoglu and Ravn, 1992, Baskerville and Myers, 2004]. It is strongly oriented towards collaboration and change involving both researchers and subjects. Typically it is an iterative research process that capitalizes on learning by both researchers and subjects within the context of the subjects' social system [Baskerville and Myers, 2004]. The essential philosophical position of action research is *pragmatism*, which concentrates on asking the right questions, and getting empirical answers to those questions [Baskerville and Myers, 2004].

There are some underlying premises under the pragmatist philosophy. Baskerville and Myers [2004] define them as follows (emphasis added):

> The first premise is Peirce's tenet that *all human concepts are defined by their consequences*. The second is James' tenet that *truth is embodied in practical outcome*. The third is Dewey's logic of controlled inquiry, in which *rational thought is interspersed with action*. The fourth premise is Mead's tenet that *human action is contextualized socially, and human conceptualization is also a social reflection*.

According to Baskerville and Myers [2004], action research is grounded in a simple two-stage process: the *diagnostic stage* and the *therapeutic stage*. First, the diagnostic stage involves a collaborative analysis of the social situation by the researcher and the subjects of the research. Theories are formulated concerning the nature of the research domain. The therapeutic stage involves collaborative change. In this stage, changes are introduced and the effects are studied.

Action research has been accepted as a valid research method in other applied fields such as organization development and education, as well as Information Systems research [Baskerville and Myers, 2004]. Good examples of action research in the software development process field can be found in [Iversen *et al.*, 2004] and, specifically in agile methods, in [Salo, 2006], [Kettunen, 2009], and [Moe, 2011]. There are also studies in team performance that adopt the action research approach (e.g., [Munro and Laiken, 2003]).

The major strength of Action Research is the in-depth and first-hand understanding the researcher obtains. Its weakness is the potential lack of objectivity on the part of the researchers when they attempt to secure a successful outcome for the client organization [Sjøberg *et al.*, 2007]. Action Research differs from case study research in that the action researcher is directly involved in planning organisational change [Avison *et al.*, 2001]. However, as discussed in Section 4.1.2, action research can be considered as a special case of case studies, since they have many properties in common.

Cycles of Action Research

There are many different ways to organize the steps and iterations in action research [Baskerville and Wood-Harper, 1998]. Table 4.3 summarizes some well-recognized action research processes. Susman and Evered [1978]'s classic action research process emphasizes diagnosing, action planning, action taking, evaluating, and specifying learning. Davison et al. [Davison *et al.*, 2004] formalize research associated with this iterative, rigorous, and collaborative process-oriented model as canonical action research. Checkland [Checkland, 1991] references Susman and Evered and emphasizes the role of the Framework and Methodology, in which research lessons are to be expressed. As a consequence, he adds an explicit exit from the iterations (step 6) and an activity in which the researcher reflects on the experience to record learning (step 7). McKay and Marshall (2001) also reference Susman and Evered, as well as Checkland. They argue that action research involves a problem solving cycle and a research cycle, that are interrelated and that often merge [Iversen *et al.*, 2004].

The five cyclical phases originally proposed by Susman and Evered [1978] are:

1. *Diagnosing* refers to the joint (researcher and practitioner) identification of actual problems and their underlying causes. During this phase, researchers and practitioners jointly formulate

Table 4.3: *Overview of Action Research Processes (adapted from Iversen et al. [2004])*

	(Susman and Evered 1978)	**(Checkland 1991)**	**(McKay and Marshall 2001)**	**Iversen (2004)**
Initiating	Establish the client-system infrastructure	1. Enter problem situation	1. Identify: problem and research theme 2. Reconnaissance: problem context and research literature 3. Plan and design: problem solving and research questions	1. Appreciate problem situation 2. Study literature 3. Select solution approach
Iterating	1. Diagnosing 2. Action planning 3. Action taking 4. Evaluating 5. Specifying learning	2. Establish roles 3. Declare framework (F) and methodology (M) 4. Take part in change process 5. Rethink 2-4	4. Action steps 5. Implement 6. Monitor: problem solving and research 7. Evaluate in terms of problem alleviation and research questions 8. Amend plan based on 7	4. Develop solution framework 5. Design solution process 6. Apply approach 7. Evaluate experiences
Closing		6. Exit 7. Reflect on experience and record learning in relation to F, M, and problem situation	9. Exit, if: problems alleviated and research questions resolved	8. Exit 9. Assess usefulness 10. Elicit research results

a working hypothesis of the research phenomenon to be used in the subsequent phases of the action research cycle.

2. *Action planning* is the process of specifying the actions to improve the problem situation.

3. *Action taking* refers to the implementation of the intervention specified in the action planning phase.

4. *Evaluating* entails the joint assessment of the intervention by practitioners and researchers.

5. *Specifying learning* denotes the on-going process of documenting and summarizing the learning outcomes of the action research cycle. These learning outcomes should contribute knowledge to both theory and practice, but they are also recognized as a temporary understanding, which serves as the starting point for a new cycle of inquiry.

Figure 4.2 illustrates the evolving spiral of continuous action research cycles.

Threats to the validity of action research

The main threat to validity for Action Research is the lack of impartiality on the part of the researcher. As researchers are so involved in the process of change, their report of the results can be

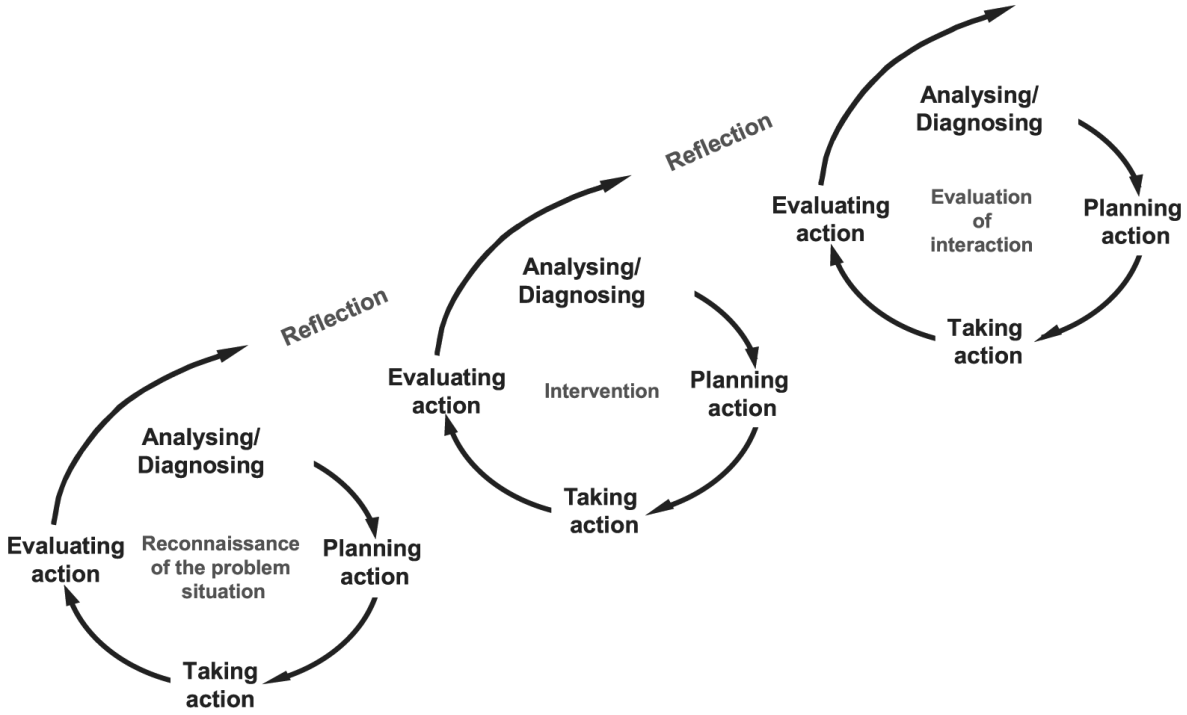

Figure 4.2: *Evolving spiral of continuous action research cycles [Vignali and Zundel, 2003]*

considered biased. Threats to the validity of action research are usually referred as *uncontrollability*, *contingency*, and *subjectivity*, and seem to be associated with the emergent nature of most Action Research investigations [Kock, 2004]. Kock describes these three threats to the validity as follows:

Uncontrollability. The essence of the uncontrollability threat is that, while the environment being studied will often change in ways that have been predicted by the researcher, sometimes change will happen in ways that are completely unexpected. The change may in some cases force the researcher to revisit his or her methods, theoretical assumptions, and even his or her research topic before a single iteration of the Action Research cycle is completed. Also, the researcher may be forced to abandon the research site before the study is completed due to events that are outside of his or her sphere of control.

Contingency. The contingency threat comes from Action Research's inherent obstacles to isolation of evidence related to particular effects and constructs from the contextual "glue" in which they are naturally found. Contingency here is used as synonymous of difficulty to generalize research findings, or difficulty to apply the research findings in contexts different from the one in which they were generated. That is, highly contingent findings carry little external validity.

Subjectivity. The subjectivity threat hinges on the fact that, in Action Research, the personal involvement of the researcher is likely to push him or her into interpreting the research data in particular and potentially subjective ways, and that, as a result, some of these interpretations may end up being completely wrong.

In order to help assure both the rigor and the relevance of Action Research in information systems, Davison *et al.* [2004] proposed five principles and associated criteria based on checklists. The principles are:

1. Principle of the Researcher–Client Agreement;
2. Principle of the Cyclical Process Model;

3. Principle of Theory;
4. Principle of Change through Action; and
5. Principle of Learning through Reflection.

We detail the checklist in Section D.3.4.

4.2 Warm-up studies (Phase I)

In the Research Phase I, we performed two main activities: i) we conducted a comprehensive literature review on agile methods, productivity definition and factors, and measurement; ii) we visited a company to conduct the first studies on the impact of agile methods in software productivity.

The main goal of the *Literature review* activity was to develop a deep background in software productivity to be able to identify major gaps in the field. The literature review resulted in Chapters 2 and 3 in this thesis, and it was also presented in papers and industry reports already published or submitted (see Chapter 1 for the list of publications).

Case studies, as we explained in the previous sections, are useful to explore new topics. In our first studies, that we here call "warm-up" studies, we adopted the case study approach. We did not consider their results as answers for the main Research Questions enunciated in Chapter 1, because they explored the broader research question: "Do agile methods increase software productivity?". However, throughout the studies, we noticed that this research question was too ambitious and context-dependent. It would be hard to answer it in the scope of a single Ph.D. Thus, we decided to narrow our research questions and then start new empirical studies to answer them.

We can summarize the design of the warm-up studies as a multiple embedded case study, since we analyzed two projects in one company and many units of analysis. We used many sources of evidence, such as surveys to employees, interviews, observation, and the company's historical database with productivity and quality software metrics. The goal was twofold: i) verify whether agile methods have positively impacted the company's productivity and product's quality, and ii) explore the relationships among agile methods, learning, and productivity.

4.2.1 Results

We conducted two exploratory case studies on agile methods impact on productivity. As mentioned before, we conducted a preliminary analysis of the effects of agile methods on software productivity of Company A. Both case studies were performed in a Brazilian government organization that adopted agile methods after a long time of using traditional software development techniques.

The first study describes the organizational context that motivated and supported this change, describes two pilot projects and discusses the observed results from technical and management perspectives. The results after 18 months were satisfactory and decisive to encourage new experiences with agile methods within the organization. First, productivity measured as function points per hour increased. Quality metrics, including the number of successful unit tests executed, and code coverage, also increased. We also conducted a questionnaire, collecting personal opinions regarding the effects of agile methods on productivity and quality. Most respondents said that agile methods significantly increased customer satisfaction, quality, and productivity.

In the second study, we explored the developers' perception of the influencing factors on their learning processes. The main results were that three factors related to XP practices can facilitate learning, potentially affecting productivity levels. Agile methods adoption seems to accelerate the process of learning new technologies. The empirical evidence also suggests that not only learning has increased, but also teams productivity. We also identified potential enabling factors of learning acceleration (three of them related to XP practices): previous professional experience, communication, constant feedback, and learning-by-doing.

However, the results were very preliminary and raised many validity threats. There were not similar agile projects to compare with at that time, just past projects developed in plan-driven approaches. Productivity definitions were based on traditional software development approaches, raising questions on the validity of the metrics adopted by the company to evaluate agile projects. Thus, the comparison between traditional and agile projects did not seem fair.

Nevertheless, the studies helped us to perform better productivity studies in the context of agile methods. We also gathered data of Company A's context, which was helpful in the next case studies. Papers P1 and P2 (Section 1.5) bring the major results from our warm-up studies.

4.3 Summary and the Challenges of this Thesis

This chapter has discussed case studies, surveys, and action research, and their respective validity threats. Some challenges in empirical research were discussed in the previous sections. This section describes which of these challenges are the subjects of this thesis in the context of agile team productivity evaluation. The Research Challenges (RCs) can be defined as:

RC1. Combing research methods and dealing with their limitations. We chose three different research approaches to answer our three research questions in an appropriate way. We should find a way to synthesize the results and provide a higher level answer based on each partial result.

RC2. Managing risks of partnership with industry. Both case studies and action research depends on the companies' commitment to enable appropriate data collection and validation. However, not always the partnerships are successful, which in turn threatens the research progress.

RC3. Interpretation. Interpreting qualitative data is one of the major threats to the validity of interpretive studies. Thus, we need to triangulate different data sources, and more important, different researchers' point of view to yield reliable results.

Chapter 5

Importance of productivity for agile teams

This chapter summarizes the results of our research according to the main findings related to the importance of productivity for practitioners adopting agile methods and perceived productivity benefits from agile methods adoption. We also analyzed relationships between companies experience, size, adopted practices, and perceived productivity.

5.1 Research method - Survey

In order to partially answer **RQ1: How important is productivity for companies adopting agile methods and how do they define productivity?**, we chose the *survey* method approach, because the first question is about *how much*. We performed a cross-sectional survey (see Section 4.1.1) in Brazil to gather quantitative data regarding the agile state-of-the-practice.

We created a web-based survey[1] consisting of 19 questions, most of which were based on a previous global survey on agile methods conducted by VersionOne [2010]. This reliance on standard instrumentation has two important advantages [Kitchenham and Pfleeger, 2008]: i) the existing instruments have already been assessed for validity and reliability and ii) by using common instruments, it is easier to compare new results with the results of other studies. Our main goal was to take an initial step towards understanding the agile methods state-of-practice in the Brazilian IT industry. In the context of this thesis, we aimed to partially answer **RQ1**, particularly the first question "How important is productivity for companies adopting agile methods?". In this phase, our focus was to explore how important is productivity for companies adopting agile methods.

Table 5.1 presents all 18 variables, scale types, and their relationship with the research subquestions. Each variable is related to one survey question, except the last question regarding the respondent contact. The survey uses a likert scale structure, which requires choices between values in the scale. Section A describes our survey protocol.

5.1.1 Data collection

When conducting research based on surveys, probabilistic sampling of participants allows making inferences about population characteristics based on sample data. However, achieving a random sample of Internet users is problematic, if not impossible [Selm and Jankowski, 2006]. Thus, we used non-probabilistic sampling techniques, recommended for exploratory research [Sue and Ritter, 2007]. We combined non-probabilistic sampling techniques, such as convenience and snowball methods to draw out our survey participants. For instance, convenience methods recruit respondents from online communities and discussion forums. Snowball sampling is based on the practice of asking participants to refer someone else to the survey, and so on.

[1] https://www.surveymonkey.com/s/KX93PGZ

Table 5.1: *State-of-practice dimensions, survey variables, and scale type*

Sub-question	Survey variable	Scale Type
Personal Profile	Personal Experience with agile	Ordinal
	Current position	Nominal
	Current exposure	Nominal
Company Profile	Company IT Size	Ordinal
	Company Experience	Ordinal
	Business Domain	Nominal
	Company Location	Nominal
Company's context	Champion	Nominal
	Worries	Binary
	Reasons for adopting Agile	Ordinal
How agile methods are being adopted	Percentage of projects using Agile	Ordinal
	Percentage of Distributed teams	Ordinal
	Agile method adopted	Binary
	Agile practices adopted	Binary
What are the perceptions after agile adoption	Benefits	Ordinal
	Agile project speed	Binary
What are the main challenges when adopting agile	Causes of failed Agile Project	Binary
	Barriers to further adoption	Binary

We piloted the survey with five agile practitioners for checking its consistency and face validity [Selm and Jankowski, 2006]. Then we drew out possible survey participants from several databases, such as mailing lists, attendees of past agile conferences, and Agilcoop[2] business contacts. We sent them an email invitation to participate in the survey, and also invited their business contacts. We collected data over a six-month survey period.

5.1.2 Data analysis

The survey data analysis was performed by one researcher (me) and revised by two researchers (me and my advisor). Since the sample is not probabilistic, we cannot statistically generalize the results. However, we can perform statistical analyses to explore data and generate propositions. Thus, we prepared the data to perform three statistical analysis: descriptive analysis of all variables, Chi-square measures of association between categorical variables and contingency tables, and Spearman's Rank Order correlation between the ordinal variables. We adopted operational definitions provided by Cohen [1988], one of the most widely known guidelines for interpreting the magnitude of correlation coefficients.

We began the survey data collection in May, 2011 and finished in October, 2011. In this period, we had 471 completed responses. To understand our sample representativeness, we compared it to the distribution of Brazilian IT professionals [Softex, 2010]. Softex [2010] has investigated the Brazilian IT industry, triangulating a large body of information and data from different government agencies[3].

[2]Cooperative for agile software development composed of IME-USP professors, students, and alumni.

[3]Their main sources are the PAS (Annual Survey of Services) and PINTEC (Survey of Technological Innovation) surveys, conducted by IBGE, the Central Register of Enterprises, also maintained by IBGE, and the RAIS (Annual

Based on this mapping, Softex has proposed the PROFSS (Software and IT services formal professionals) concept. PROFSS can be IT managers; network, systems, and database administrators; computer systems designers and analysts; and related occupations, such as computer assistants and operators. We used the most recent PROFSS statistics to compare them to our sample, as shown in Table 5.1. Our sample better represents the Southeast and Midwest regions.

Table 5.2: *Distribution of Brazilian IT companies by region and Respondents' region distribution*

	Distribution of PROFSS by Region, 2005*	**Our Sample**
Region	**% IT Professionals**	**% respondents**
Southeast	56.6	58.8
South	13.6	10.4
Northeast	15.3	13.7
North	5.3	4.0
Midwest	9.2	11.6

**Softex [2010], pg. 134*

Participants profile

To characterize the participants in the survey, we collected data about their experience with agile methods, current position, and exposure to agile development. Figure 5.1 summarizes the participants profile. The results show that more than 50% of the practitioners have already moved to agile methods, having at least 1-year of experience.

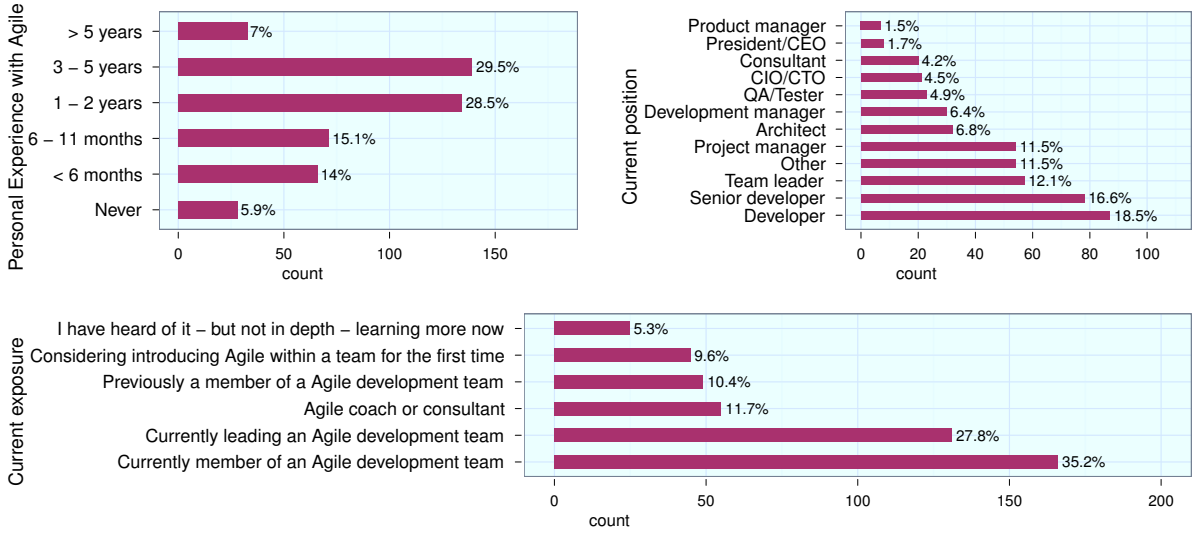

Figure 5.1: *Participants personal profile*

The participants current position (Figure 5.1) varied widely from developer to product manager roles, ensuring diversity in our survey. Developers and senior developers account for 35.1% of participants. Team leaders and Project managers account for 23.6%. This result points out that more than half of respondents play a developer or a manager role. In the option "Other", which represents 11.5% of our respondents, there are roles such as systems analyst, business analyst, requirements analyst, researcher, and trainer.

Finally, Figure 5.1 depicts participants current level of exposure to agile development. Most participants are currently members, leaders, or coaches of an agile team.

Listing of Social Information), conducted by the Ministry of Labor.

We characterized the participants companies profile by describing the company IT team size, their experience with agile methods, location, and business area. Figures 5.2 and 5.3 illustrate our results.

Most participants companies (Figure 5.2) are small or very small organizations, having an average IT department size smaller than 20 people. However, 23.6% of companies employ 21 to 100 people, and 22.7% employ more than 100 people in the IT area. Company experience in practicing agile development methods is mostly between one and two years (31.4%), and then between three and five years (24.8%). The results point out that more than half of the companies are experimenting or effectively using agile, having at least 1-year experience. Some companies indicated (10.4%) that they *never* used agile methods. We interpret them as potential future adopters that are still analyzing agile methods benefits before implementing it.

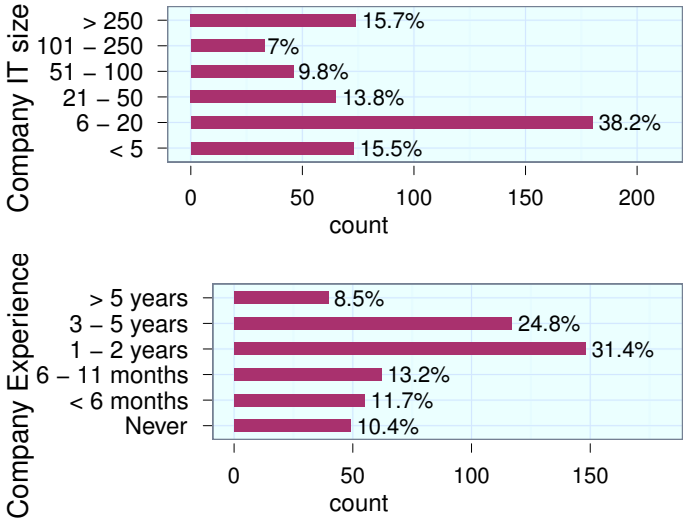

Figure 5.2: *Participant's Companies Profile - IT size and Experience with Agile Methods*

Figure 5.3 frames the organization main activity. Most of them are related to the Internet, Government, and Office/Business areas. In the "Other" option, many of them are associated to software factories, information technology (IT) in general, and research and development (R&D) segments. Figure 5.3 also presents the participants organizations location. São Paulo, Rio de Janeiro, Distrito Federal, and Minas Gerais host most of the companies.

5.2 Empirical results

Our objective is to provide a better understanding of the importance of productivity for agile teams and potential productivity benefits obtained from agile methods adoption. We describe, below, results from the survey analysis that support our findings.

5.2.1 Key finding 1: Productivity as an important reason for adopting Agile Methods

Our survey describes the companies context when adopting agile methods, focusing on who was the champion, the main reasons they find relevant for adopting agile development methods, and what worries companies have at that time. Figure 5.4 summarizes the main findings.

Despite the results show different aspects of agile methods adoption, we will focus on explaining productivity-related data. Figure 5.4 illustrates the main reasons that led companies to adopt agile methods. The graphic sums up, in the left side, the frequency of "Highest importance" and "Very important". In a similar way, the right side presents the sum of frequencies of "Somewhat important" and "No important at all". Our results show that the most important reason for companies to

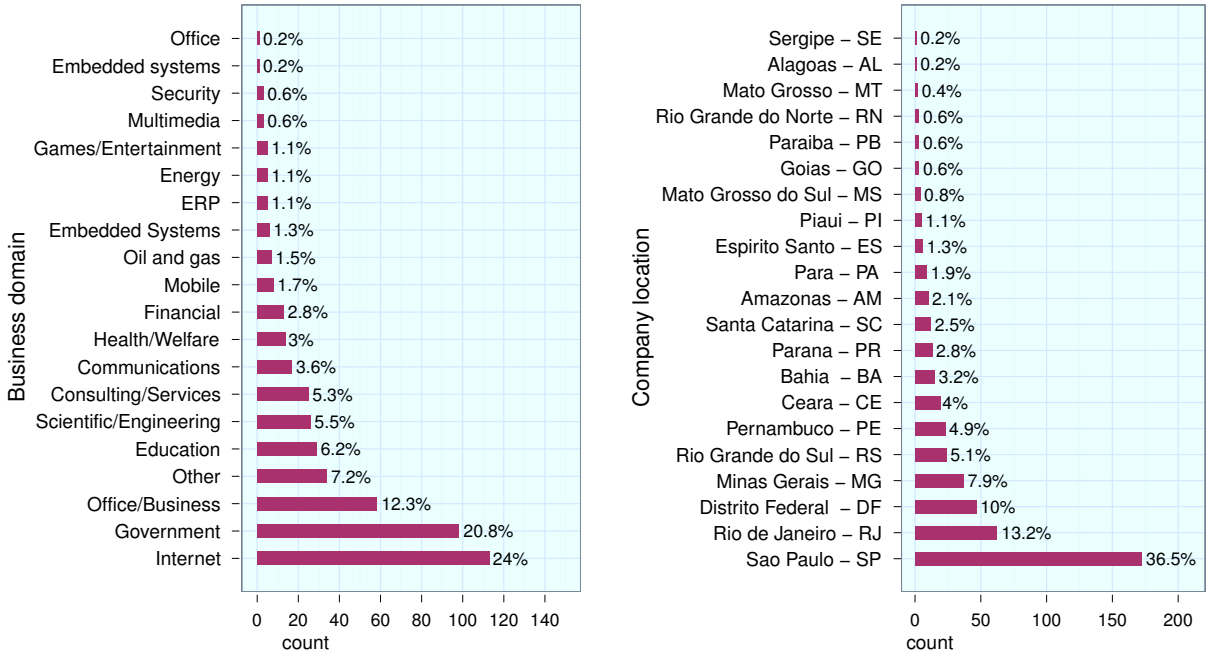

Figure 5.3: *Participants Companies Profile - Location and Business Area*

adopt agile methods was to *Increase productivity* (91%). *Manage changing priorities* (86%) and *Enhance software quality* (83%) are other important reasons, according to our respondents. There were other expectations, such as *Simplify development process* and *Accelerate time to market*. The reasons, therefore, are aligned with the agile mentors claims [Beck, 1999, Schwaber and Beedle, 2001]. Surprisingly, 53% of respondents do not considered *Reduce cost* as an imperative reason to adopt agile methods.

5.2.2 Key finding 2: Productivity as one of the perceived benefits from adopting Agile methods

With agile adoption, organizations may benefit in many ways, Figure 5.5 discloses the perceived benefits. The heat map illustrates trends about companies' perceptions. Our result shows that *Productivity* (69.2%), *Ability to manage changing priorities* (67.9%), *Team morale* (66.8%), *Simplified development process* (60.9%), and *Quality* (60.2%) have improved or significantly improved after the adoption of agile methods. The bottom-ranked benefit was the ability to *Manage distributed teams* (24.8%). However, most respondents do not experience distributed development, which diminishes this result relevance.

Regarding the effective project speed obtained from implementing agile methods, as shown in Figure 5.6, most respondents (67.1%) indicated that agile projects provide faster time to completion. For 13.2% of respondents, agile projects have the same time to completion. A smaller group (0.8%) found agile projects slower to complete. Finally, the other respondents (18.9%) have not yet finished an agile project to answer this question. The overall result points out that agile methods leads to better project speed.

5.2.3 Key finding 3: Perceptions of productivity are not associated with the company size nor experience with agile methods

Considering that productivity was the most important reason to adopt agile methods and one of the most perceived benefits of agile methods, we decided to evaluate if this result was related to some company properties, such as size or experience with agile methods. A Spearman's Rank Order correlation was executed to determine the relationship between reasons for agile methods adoption, the perceived benefits, and company size and experience, as shown in Table 5.3.

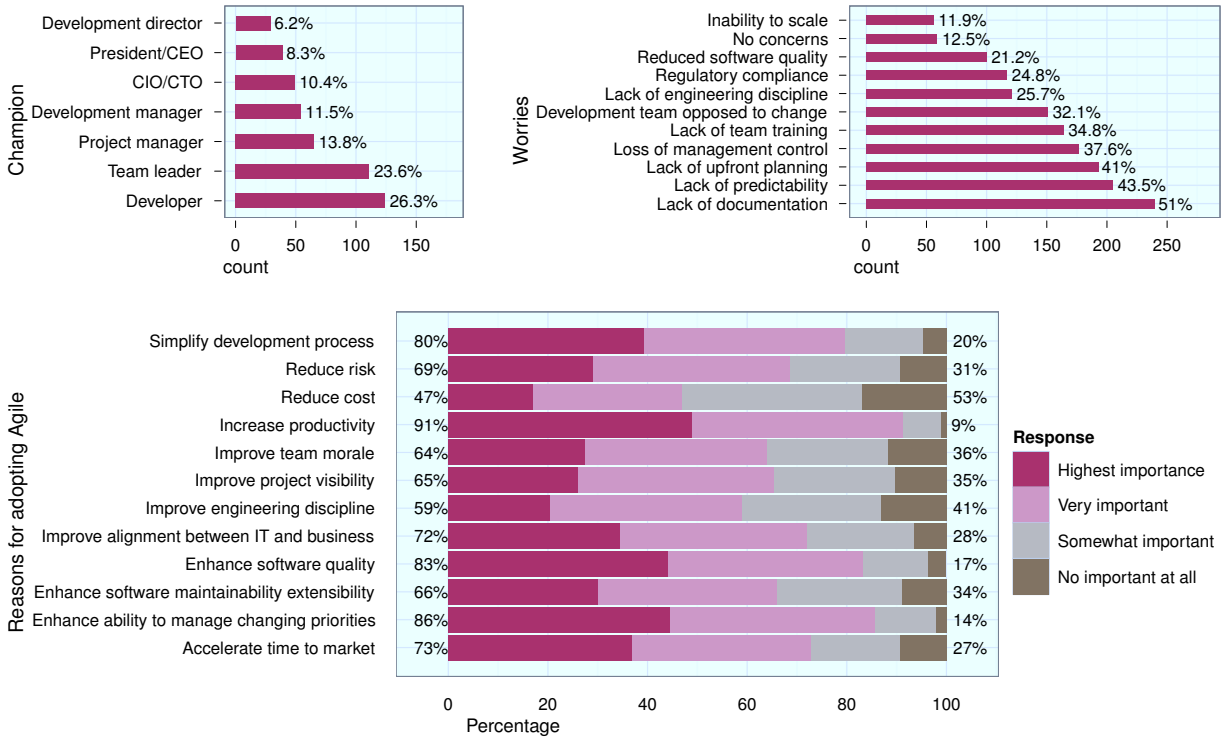

Figure 5.4: *Champions, Worries, and Reasons for adopting Agile*

There was a very weak correlation between the reasons that led to agile adoption and the company size or experience with agile methods. For instance, there is a negative weak correlation between *Company size* and *Maintainability* (r =- 0.017, $\rho < 0.001$). The overall result denotes that reasons for agile adoption are not associated with the company size nor experience with agile methods.

There was a moderate, positive correlation between the *perceived benefits* (*visibility*, *quality*, *cost*, *change management*, *time-to-market*, *risk management*, and *alignment between business and IT*) and *company experience*, which was statistically significant (see Table 5.3). Thus, these perceived benefits tended to be higher for more experienced agile companies.

Thus, in both analyses, productivity is not related to the companies sizes and experience with agile methods. This result shows that productivity is the most important reason to adopt agile methods irrespective of the company size or experience. Agile methods also tend to generate a positive perception of productivity increasing, both in large, medium, and small companies, as well as inexperienced and experienced companies.

Our results also point out a strong correlation among the most perceived benefits, as shown in Table 5.3. This shows that benefits tend to be achieved together.

5.2.4 Key finding 4: Agile practices adopted by companies perceiving significantly improved productivity

To evaluate agile practices, we borrowed a list of practices from surveys carried out by VersionOne [2009, 2010] due to their large impact running worldwide agile surveys. Figure 5.7 shows that *Iteration planning*, *Retrospectives*, *Unit testing*, *Daily standup*, and *Refactoring* are the five most adopted practices, according to our respondents. These practices represent management, continuous improvement, quality, and architectural valued aspects in teams.

We then analyzed companies that significantly improved productivity after adopting agile methods and the practices they are applying. Figure 5.8 shows the analysis results. In companies observing productivity increasing, the most adopted practices are again *Iteration planning*, *Retrospectives*, *Unit testing*, *Daily standup*, and *Refactoring*. This is the same result we found for all respondents,

Table 5.3: *Spearman's correlation between Reasons for Agile adoption, Company Size, and Experience*

	1	2	3	**4**	5	6	7	8	9	10	11	12	13
Company size [1]													
Company experience [2]	0.30***												
Managing priorities [3]	0.09	0.14**											
Productivity [4]	-0.04	-0.04	0.11*										
Quality [5]	-0.06	0.05	0.16***	0.19***									
Business and IT alignment [6]	0.10*	0.07	0.31***	0.07	0.26***								
Visibility [7]	0.07	0.11*	0.21***	0.16***	0.19***	0.32***							
Time to market [8]	0.10*	0.11*	0.19***	0.16***	0.04	0.14**	0.23***						
Simplify development [9]	-0.04	0.13**	0.09*	0.22***	0.23***	0.04	0.08	0.21***					
Engineering discipline [10]	-0.08	0.07	0.15***	0.15***	0.29***	0.13**	0.24***	0.11*	0.33***				
Cost [11]	-0.10*	0.05	0.12*	0.23***	0.20***	0.18***	0.18***	0.18***	0.25***	0.25***			
Maintainability [12]	-0.17***	0.02	0.19***	0.19***	0.52***	0.25***	0.17***	0.09*	0.27***	0.36***	0.38***		
Team morale [13]	-0.04	0.04	0.16***	0.14**	0.24***	0.20***	0.30***	0.11*	0.22***	0.30***	0.17***	0.34***	
Risk management [14]	-0.02	0.14**	0.29***	0.11*	0.23***	0.30***	0.30***	0.07	0.17***	0.22***	0.29***	0.28***	0.43***

Table 5.4: *Spearman's correlation between Perceived Benefits adopting Agile, Company Size, and Experience*

	1	2	3	**4**	5	6	7	8	9	10	11	12	13	14
Company size [1]														
Company experience [2]	0.30***													
Visibility [3]	0.21***	0.33***												
Productivity [4]	0.10*	0.29***	0.67***											
Quality [5]	0.11*	0.32***	0.68***	0.76***										
Cost [6]	0.12**	0.31***	0.58***	0.71***	0.67***									
Simplify development [7]	0.12*	0.29***	0.64***	0.72***	0.70***	0.68***								
Engineering discipline [8]	0.12*	0.27***	0.65***	0.68***	0.70***	0.62***	0.70***							
Team morale [9]	0.16***	0.28***	0.68***	0.72***	0.70***	0.61***	0.70***	0.72***						
Change management [10]	0.21***	0.35***	0.71***	0.69***	0.67***	0.65***	0.69***	0.66***	0.76***					
Time-to-market [11]	0.16***	0.31***	0.65***	0.67***	0.60***	0.70***	0.67***	0.64***	0.68***	0.73***				
Risk management [12]	0.16***	0.33***	0.66***	0.69***	0.69***	0.63***	0.66***	0.67***	0.69***	0.71***	0.66***			
Distributed teamwork [13]	0.17***	0.27***	0.46***	0.44***	0.48***	0.56***	0.48***	0.50***	0.46***	0.47***	0.49***	0.51***		
Business and IT alignment [14]	0.21***	0.35***	0.65***	0.65***	0.67***	0.62***	0.65***	0.65***	0.65***	0.72***	0.67***	0.72***	0.54***	
Maintainability [15]	0.06	0.29***	0.64***	0.73***	0.78***	0.67***	0.70***	0.72***	0.68***	0.66***	0.62***	0.68***	0.53***	0.72***

*N = 471 Significance levels : ***$\rho < 0.001$ **$\rho < 0.01$ *$\rho < 0.05$*

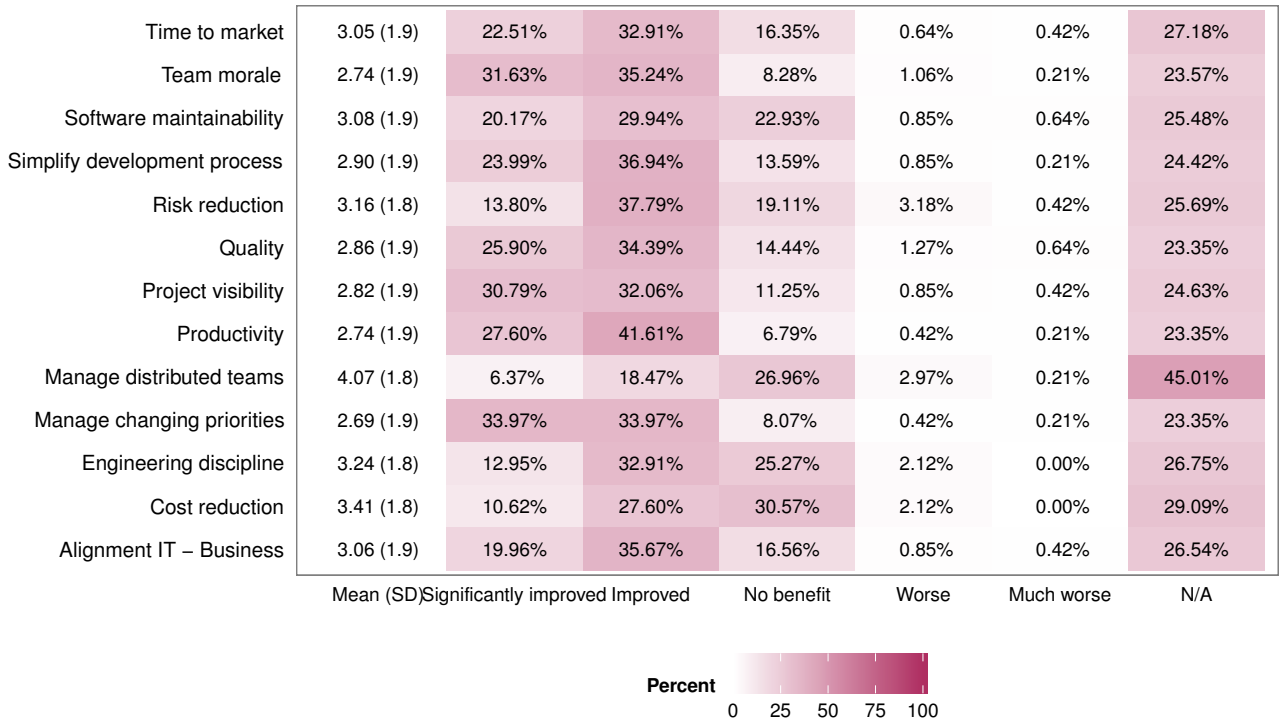

Figure 5.5: *Perceived benefits from implementing Agile practices*

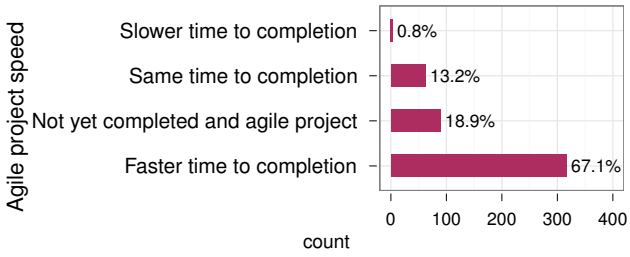

Figure 5.6: *Project speed after implementing Agile practices*

shown in Figure 5.7. Therefore, even for those who did not observe productivity improvement, the most adopted agile practices are the same. This result suggest that agile practices alone might not explain the perceived productivity increase.

5.3 Discussion

Most existing surveys on agile adoption [VersionOne, 2008, 2009, 2010, 2011] were carried out by industry representatives. They usually collect data from different countries, bringing an overall view of the state of practice of agile methods. However, these surveys do not elaborate much on how data are associated, and do not explore productivity issues as well.

When we compare the results between Brazilian and worldwide surveys [VersionOne, 2011], we found very similar results about the benefits gained from implementing agile methods and four of the top five practices most adopted. Most used agile practices according to our study were *Iteration planning*, *Retrospective*, *Unit testing*, *Daily standup*, and *Refactoring*.

Similar studies in other countries have suggested that most used practices are Daily stand-up meetings [Rodríguez *et al.*, 2012, Salo and Abrahamsson, 2008, VersionOne, 2011, West and Grant, 2010], Coding standards [Begel and Nagappan, 2007, Salo and Abrahamsson, 2008], Iteration planning [Rodríguez *et al.*, 2012], Continuous Integration [Begel and Nagappan, 2007], unit testing [VersionOne, 2011], Constant customer feedback [West and Grant, 2010], 40 hours week [Salo and

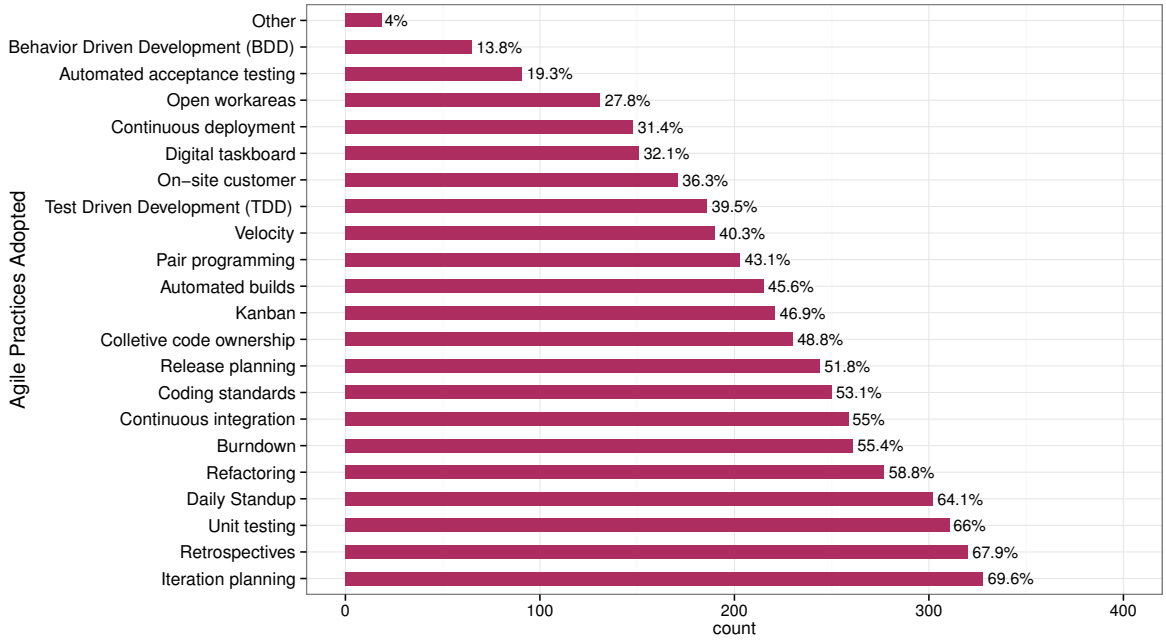

Figure 5.7: *Most adopted agile practices*

Abrahamsson, 2008], Product backlog [Rodríguez *et al.*, 2012, Salo and Abrahamsson, 2008], and Open office space [Salo and Abrahamsson, 2008]. Our results only differ from previous research on the *Retrospective* and *Refactoring* practices. It is not clear why the results differ, since data from previous studies is not available.

Our findings show that *Productivity* was the most frequent goal when adopting agile methods, and also one of the most perceived benefits from agile methods adoption, according to the survey respondents. This result supports our proposition (Chapter 2) that, despite the apparent paradox between productivity and agility, productivity is still important for agile teams. Our result also confirms previous studies that have shown productivity as the most important goal when adopting agile methods [Rodríguez *et al.*, 2012, VersionOne, 2010].

We also investigated associations between company size, experience with agile methods, and perceptions on productivity (productivity as a reason and as a benefit). Our results show there is no correlation between these variables. This implies that both reasons and benefits do not tend to be higher for more experienced agile companies, or more pronounced in big companies. There is no similar evidence on past research to contrast our results. One could imagine that smaller companies would perceive more productivity benefits than larger companies, because agile methods are usually suited to smaller groups. Our results challenge this assumption, since we observed that these variables are not correlated.

Finally, answering RQ1, productivity is not only important for teams adopting agile methods, but it is also an outcome from agile adoption. This result motivates more research on agile team productivity and is an important step to substantiate our research problem and goals.

5.3.1 Limitations

This study presents some limitations. Our main goal was to present a big picture of the agile state of the practice in Brazil, particularly focused on productivity issues. Despite the sample being representative of the distribution of IT companies in the country, it is just valid in our research context and cannot be generalized at this time.

Other limitations are related to individuals representing organizations and their roles. Since our sample was completely anonymous, we could not evaluate whether respondents work for the same company or not. This limitation might bias the sample by giving more weight to answers from

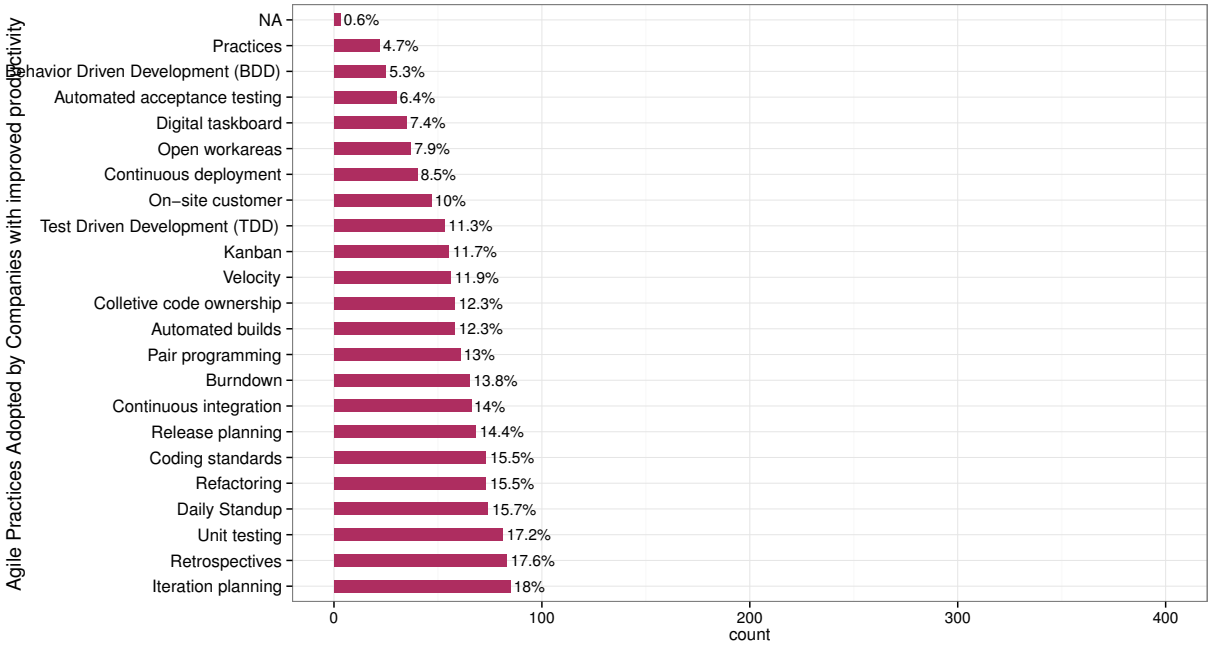

Figure 5.8: *Most adopted agile practices by companies with significantly improved productivity*

certain companies instead of covering more organizations in each Brazilian region. It would be more appropriate to analyze results at the organization level, designing the study to gather data from different organizations and make sure that the number of answers from each company are similar. Another limitation is that respondents play different roles at their companies. Different roles bring different viewpoints and knowledge depth, which might affect respondents opinions and answers about agile methods and practices.

Despite of these limitations, our results can be used in future comparisons among countries that have had similar research studies. This will increase the current knowledge on agile methods adoption and results in the industry.

5.4 Summary

This chapter presented our results which address the importance of productivity for Brazilian practitioners, aiming at partially answering **RQ1: How important is productivity for companies adopting agile methods and how do they define productivity?**.

We claim that our main contribution is **C1. Empirical verification of the importance of productivity for companies adopting agile and the perceived benefits**.

Chapter 6

Agile team productivity definition and factors

This chapter presents the conceptual framework we developed to explore agile team productivity, and describes the research method, results, and discussion of main findings related to agile team productivity definitions and factors.

6.1 Conceptual framework to study agile team productivity

We developed a conceptual framework for studying agile team productivity, which is useful to provide clarity and focus in the study, and to drive further discussion around the results [Miles and Huberman, 1994]. It incorporates all aspects discussed in this chapter, providing details on factors impacting on team productivity and how the impact can be evaluated.

Though team productivity has been studied in the software development field, the most mature theoretical models addressing teamwork components and productivity outcomes come from the organizational behavior area (e.g., Cohen and Bailey [1997], Guzzo and Dickson [1996], Marks *et al.* [2001], Mathieu *et al.* [2008], Salas [2005], Yeatts and Hyten [1998]). According to Salas *et al.* [2004], *teamwork* is "a set of interrelated thoughts, actions, and feelings that combine to facilitate coordinated, adaptive performance and the completion of taskwork objectives". Software development, especially agile software development, relies predominantly on teamwork [Tang and Kishore, 2010]. This literature is, therefore, relevant for studying agile team productivity.

One well-known theoretical model for teamwork effectiveness is the *Input - Process - Output* (IPO). In this framework, team effectiveness is a function of input factors and group processes, where both team inputs and processes have important and differentiated impacts on team performance [Faraj and Sproull, 2000]. In software development, the IPO model has been applied to analyze the impact of team input (e.g., team development stage, team characteristics) and processes (e.g., coordination, communication quality, knowledge sharing) on team effectiveness, performance, and other outcomes as software quality and job satisfaction [Acuña *et al.*, 2009, Faraj and Sproull, 2000, Lu *et al.*, 2011, Stewart and Gosain, 2006]. In agile development, the IPO model (and its evolution) has recently been adopted as an underlying conceptual framework in both quantitative and qualitative studies (e.g., Tang and Kishore [2010], Whitworth and Biddle [2007]).

In our work, we adapted three IPO teamwork effectiveness frameworks from Cohen and Bailey [1997], Yeatts and Hyten [1998], and Marks *et al.* [2001], aiming to describe a more coherent conceptual framework of agile team productivity. Based on these frameworks, we selected inputs, processes, and outcomes related to the agile values and principles [Beck *et al.*, 2001]. Figure 6.1 presents the novel conceptual framework we use to support the agile team productivity data analysis. In this new conceptual framework, we classify input into five subgroups (I1-I5), one subgroup to explain group processes (G1), and two subgroups to explain productivity outcomes (O1 and O2). We describe them bellow.

Input factors. *Individual and Group characteristics (I1)* describe member and team types. Most theoretical team performance frameworks have included team design characteristics, including team size and composition [Yeatts and Hyten, 1998]. Teams sufficiently well designed to perform adequately are more likely to be given additional authority over their work, more supporting resources, and more challenging goals [Wageman, 2001]. According to Bell [2007], team design could be related to team performance, as it affects the amount of knowledge and skills that team members must apply to the team tasks.

The most relevant team member characteristics are knowledge, skills, motivation, and personality, while team characteristics include size, diversity, staff turnover, and shared beliefs. Team capabilities and skills are the most significant personnel characteristics that influence software productivity [Maxwell and Forselius, 2000, Tan *et al.*, 2009] and usually play a moderator role in frameworks that aim to explain productivity variations [Blackburn *et al.*, 2000, Trendowicz and Münch, 2009]. Agile development is essentially people-centric and recognizes the value of team member competencies when bringing agility to development processes [Lee and Xia, 2010, Nerur and Balijepally, 2007]. Getting the right people with appropriate skills and empowering them are critical for agile development success [Chow and Cao, 2008, Highsmith, 2004]. Team diversity is key for agile development [Nerur and Balijepally, 2007], being an XP principle [Beck and Andres, 2004]. Teams with broader experience are positively associated with project performance [MacCormack *et al.*, 2001]. These claims suggest that group characteristics are an important factor impacting on agile team productivity.

Stage of team development (I2) relates to team maturity. Team members must learn new behaviors and skills to improve their work and create a high-performance team. Several models describe the team development stage, such as the Tuckman [1965] classic model of forming, storming, norming, and performing. In the forming stage, team members attempt to create social and task structures to guide their interactions. When they realize that it is difficult to create consensus on a certain approach, they shift to a storming stage, in which different members compete for influence. The team evolves and reconciles differences by setting norms to guide their interactions. Once the norms are well established, members can focus on achieving common goals. As agile methods focus on teamwork, which varies according to the development stage, we included this subgroup in the framework.

Nature of task (I3) includes task design, task duration, the degree of autonomy to execute the tasks, and task interdependencies. Cohen and Ledford [1994] argue that enriched tasks allow variety, significance, autonomy, and feedback, which result in high responsibility, motivation, satisfaction, and team performance. In software development, proper task assignment is clearly considered to impact productivity [Boehm, 1981], because it can influence team member motivation [Procaccino *et al.*, 2005]. Agile teams should have sufficient autonomy to determine which tasks must be performed, demonstrating results at the end of each iteration [Augustine *et al.*, 2005]; we thus included this subgroup in our conceptual framework.

Organizational context (I4) includes variables such as rewards, culture, training, and resources. Collective rewards help motivate groups whose tasks were made interdependent, while individual rewards acknowledge members whose performed tasks reflect individual responsibilities [Cohen and Bailey, 1997]. Several agile teams (including those studied here) work within an organizational environment. We thus included this subgroup as input.

Supervisory behaviors (I5) rely on leadership style – whether it is transactional or transformational and whether it guides the team directly or encourages self-management. Transactional leaders usually set goals, obtain team agreement on what is to be accomplished, and monitor team performance. Transformational leaders are inspiring and stimulating, providing followers with a sense of purpose, articulating shared goals and mutual understanding, and an attractive future. To do so, they consider the maturity level, capabilities, and subordinates' needs by treating employees as unique individuals [Flin and Yule, 2004]. We added *Supervisory behaviors* because self-management and empowerment are considered key for agile development, supported by agile practices and essential for agile culture [Highsmith, 2004, Kelly, 2008, Sharp and Robinson, 2004].

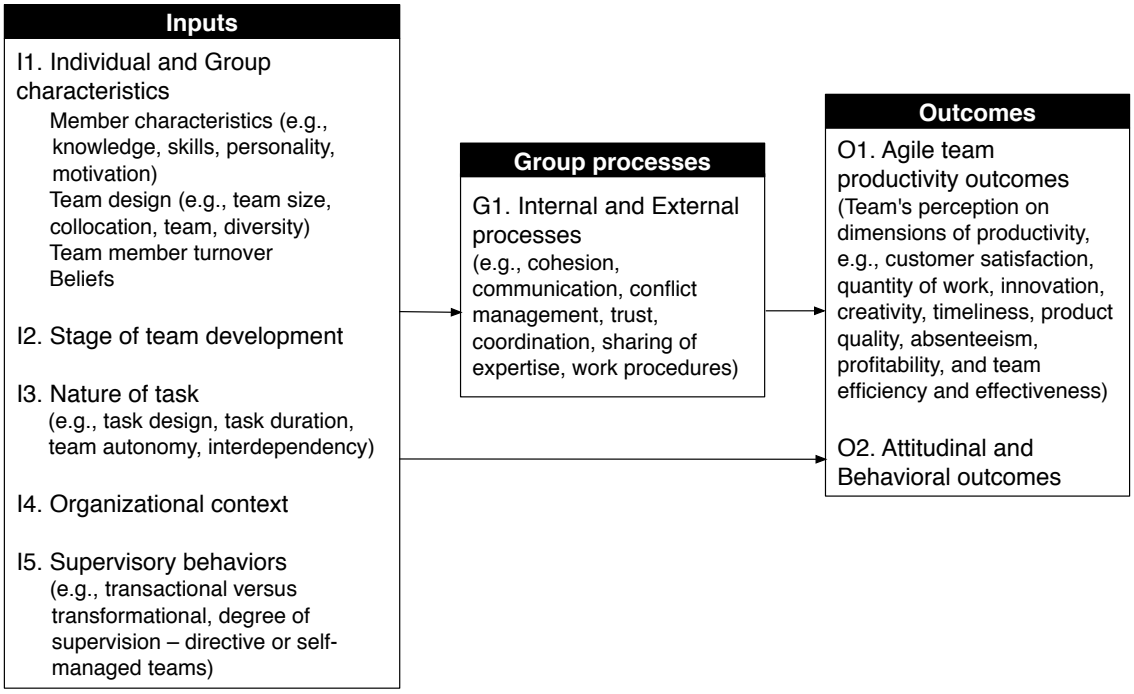

Figure 6.1: *Our agile team productivity conceptual framework*

Group processes (G1) Interactions among team members and interactions with other teams, customers, and suppliers directly affect team performance [Yeatts and Hyten, 1998]. Group processes also mediate the relationship between inputs and outcomes. Team interpersonal processes and work procedures are considered group processes. Examples of group processes are team cohesion, team communication, trust, conflict management processes, and how they coordinate their activities (coordination processes). Moreover, agile methods and their practices are work procedures played by team members that may affect productivity directly or, at least, mediate the relationship between input factors and productivity outcomes. Because agile methods focus on people, teamwork, and their interactions through agile practices, all those processes may have a significant influence on team productivity and were included in our framework.

Outcomes (O1 and O2) As output, there are some expected outcomes, including *agile team productivity (O1)* and *attitudinal and behavioral indicators (O2)*. As productivity is hard to measure (Chapter 3, Section 3.2), we considered agile team productivity as the team's own perception of their overall productivity. Considering that team's perception may vary substantially over time, we added all knowledge worker productivity dimensions as possible outcomes in the model. Attitudinal and behavioral outcomes (e.g., trust and commitment) were included because of their importance in establishing agile teamwork and self-organization [Moe *et al.*, 2010, 2008].

6.2 Research method - Multiple-Case Studies

We aimed at partially answering **RQ1. How important is productivity for companies adopting agile and how do they define productivity?**. In this phase, our focus was to explore how do companies define productivity when adopting agile methods. We also aimed to explore **RQ2: What factors impact agile team productivity and how is this impact from the team point of view? Which agile practices are perceived to impact on a given team's productivity?**.

To answer these research questions, we performed a multiple-case study in the Brazilian IT industry. Case studies [Gerring, 2006, Verner *et al.*, 2009] have high potential for analyzing perfor-

mance improvement, and they are appropriate for studying complex performance issues [Mulder, 1999].

When designing the multiple-case studies, we divided our goals into three (sub) research-questions (SQ):

- *SQ1. How do agile teams define productivity?*
- *SQ2: What do agile team members identify as the main factors impacting on productivity?*
  - *How do these factors impact positively, or negatively on the productivity of agile teams?*
- *SQ3: Which agile practices are perceived to impact on a given team's productivity?*

The criteria for case selection included the following: (1) companies using agile methods (XP [Beck and Andres, 2004] or Scrum [Schwaber and Beedle, 2001]) for at least two years; (2) companies in different business segments, geographical location, size, structure, and culture; (3) agile projects with at least four co-located developers and in progress for at least six months.

Data collection was carried out in three Brazilian companies, from September 2010 to February 2011. The unit of analysis is a set of three development projects, one in each company. We chose to follow the teams for six months because the influence of some productivity factors may change over time, depending on the project context. The staff turnover problem may be noticeable only immediately after a member's dismissal. If we collected the data in a single point in time, this event might not be mentioned during the interviews.

We signed a non-disclosure agreement with the companies; this step was important to establish a formal link between researchers and companies and ensure data confidentiality, so the companies would feel more comfortable with our presence observing their internal activities.

6.2.1 Data collection

Table 6.1 describes the company and project profiles, considering guidelines provided by Kitchenham *et al.* [2002]. Table 6.2 shows the agile practice adoption level for each project. If a team used one practice fully, we assigned the term *full*. If they used just a few recommendations of the practice, we assigned *partial*. When they did not use it, we assigned the phrase *Do not use*. We described these profiles based on teams inputs.

Company A is a large financial corporation with over 500 IT employees, which had previously used plan-driven development processes. The company managers decided to adopt agile methods to increase team productivity, and they had been using them for two years. The organizational structure and coordination are primarily vertical [Galbraith, 1973], where project managers usually implement coordination processes. Project A is a re-development of an existing system for the financial market involving several institutions. The project started in March 2010 and was estimated to last for approximately two years. The team adopted several XP and Scrum practices and used one-week iterations.

Company B has been delivering e-commerce and infrastructure services for over ten years and has used only agile methods to develop software. It employs approximately 120 developers. The organizational structure and coordination are primarily horizontal [Galbraith, 1973], where coordination processes are usually provided by an individual team member who communicates directly with other members or users on a one-to-one basis. Project B is a new development of an e-commerce service in a market with other competitors. The project also started in March 2010 but does not have a specific deadline, as they are developing software as a service, with continuous improvement and new functionalities. The project adopts several XP, Scrum, and Lean principles and practices.

Company C is an important player in Internet content and access provision in Brazil. The organizational structure and coordination are primarily vertical [Galbraith, 1973], but the hierarchy is smaller than Company A. The IT department employs approximately 200 developers and had also previously used plan-driven development processes. They have applied agile methods since 2008. Project C is the maintenance of a recommendation system for products from several virtual stores. The project adopts mainly Scrum practices and some XP and Lean principles and practices.

Table 6.1: *Company and Project Profiles*

Characteristics	Company/Project A	Company/Project B	Company/Project C
Company business	Financial	E-commerce and Infra-structure services	Internet content and provider
Company structure	Vertical	Horizontal	Vertical
Number of IT employees	400	120	200
Project description	Financial system	E-commerce service	Recommendation system
Team composition	6 full-time developers, 2 part-time developers, 1 scrum master, 1 project manager, 1 product owner (Total = 11 members)	4 full-time developers, 2 part-time developers, 1 project manager/coach, 1 product owner (Total = 8 members)	4 full-time developers, 1 webmaster, 1 test specialist, 1 scrum master, 1 project manager, 1 product owner (Total = 9 members)
Language	Java	Ruby	Java
Non-functional requirements	Reliability, Availability, Performance	Reliability, Availability	Performance, Availability and Auditability
Reuse	High - Software product lines, components and other systems	High - Open source project and other systems	High - Components and other systems
Requirements stability	High stability	Medium stability	Medium stability
Staff turnover	33.3% - Considered medium by the project manager	40% - Considered medium by the project manager	35.3% - Considered medium by the project manager

Data collection instruments

The main data collection methods were semi-structured interviews, non-participant direct observations, face-to-face discussions with project leaders, and document analysis (Table 6.3). The data were collected over a period of 6 months, gathering opinions from different stages of the project. Since we played different roles in the data collection and analysis, from now on we make a distinction between researchers to make each one's participation clearer.

Interviews were semi-structured (Appendix B provides the interview guide) to understand the factors impacting project productivity in the team's perception and how they impacted. I conducted the interviews. I have experience conducting interviews due to my background in requirement elicitation in real projects. Each interview lasted approximately one hour, and the interviewees were informed about the audio recording and its importance to the study.

We conducted interviews with 19 team members within the 3 companies, including developers, project managers, and product owners, also considering different experience profiles. We informed all participants of the main research goal but did not give further details, which could have biased their opinions on the research subject.

We developed a protocol (Appendix B) to guide the non-participative observation and register observations about factors impacting team productivity and any other exceptions during the project or team follow-up. The protocol contains questions answered regularly by the observer (daily or per iteration), role that I performed. We also collected retrospective documentation – all the teams maintain such information in spreadsheets or wiki. Table 6.3 summarizes the data sources used in the study.

Table 6.2: *XP and Scrum practices adopted by the projects*

Practices	Project A	Project B	Project C
Code & Tests	Full	Full	Partial
Continuous integration	Full	Full	Partial
Daily deployment	Do not Use	Partial	Partial
Daily meeting	Full	Full	Full
Energized work	Partial	Full	Partial
Incremental design	Full	Full	Full
Pair programming	Full	Partial	Partial
Real customer involvement	Full	Full	Full
Shared Code	Full	Full	Full
Single code base	Full	Full	Full
Sit together	Partial	Full	Full
TDD	Partial	Full	Partial
Ten minute build	Full	Full	Partial
Negotiated scope contract	Do not Use	Partial	Partial
Planning game	Full	Partial	Full
Retrospectives	Full	Full	Full
Root cause analysis	Do not Use	Full	Do not Use
Slack	Full	Full	Full
Stories	Full	Full	Full
Team continuity	Partial	Partial	Partial
Weekly cycle	Full	Partial	Partial
Whole team	Partial	Partial	Partial

Finally, agile teams are composed of knowledge workers (KW) working together and sharing the same goal. The product of a KW is typically intangible: knowledge is the addition of meaning, context, and relationships to data or information [Bosch-Sijtsema *et al.*, 2009]. Even if there are no universally accepted methods to measure KW productivity, it has been studied through its dimensions, including a KW's perception of his productivity, customer satisfaction, quantity of work, innovation, creativity, timeliness, product quality, absenteeism, profitability, and team efficiency and effectiveness [Ramírez and Nembhard, 2004]. However, most companies measures productivity in different ways, which makes a comparison impossible. In this study, we have analyzed agile team productivity using the team's perception as one potential dimension to understand their overall productivity. Through perceptions, we were able to establish a single dimension for the three different companies.

6.2.2 Data analysis

We used thematic analysis to analyze the data, a technique for identifying, analyzing, and reporting standards (or themes) found in qualitative data [Boyatzis, 1998, Braun and Clarke, 2006, Cruzes and Dybå, 2011]. It is a way to recognize patterns in textual data, where emerging themes become categories for analysis [Fereday and Muir-Cochrane, 2006]. To support the data analysis,

Table 6.3: *Description of the data sources in our study (adapted from Yin [2008])*

Data source	Company/Project A	Company/Project B	Company/Project C
Retrospective documentation	27 (1-week) iterations	15 (3 or 4-week) iterations	18 (3-week) iterations
Interviews	3 full-time developers, 1 part-time developer, 1 product owner, 1 scrum master, 1 project manager Total = 7	1 project manager/ coach, 1 product owner, 4 developers Total = 6	3 full-time developers, 1 test specialist, 1 webmaster, 1 scrum master Total = 6
Direct observation	Visiting the open office spaces. Field notes taken in the regular work sessions		

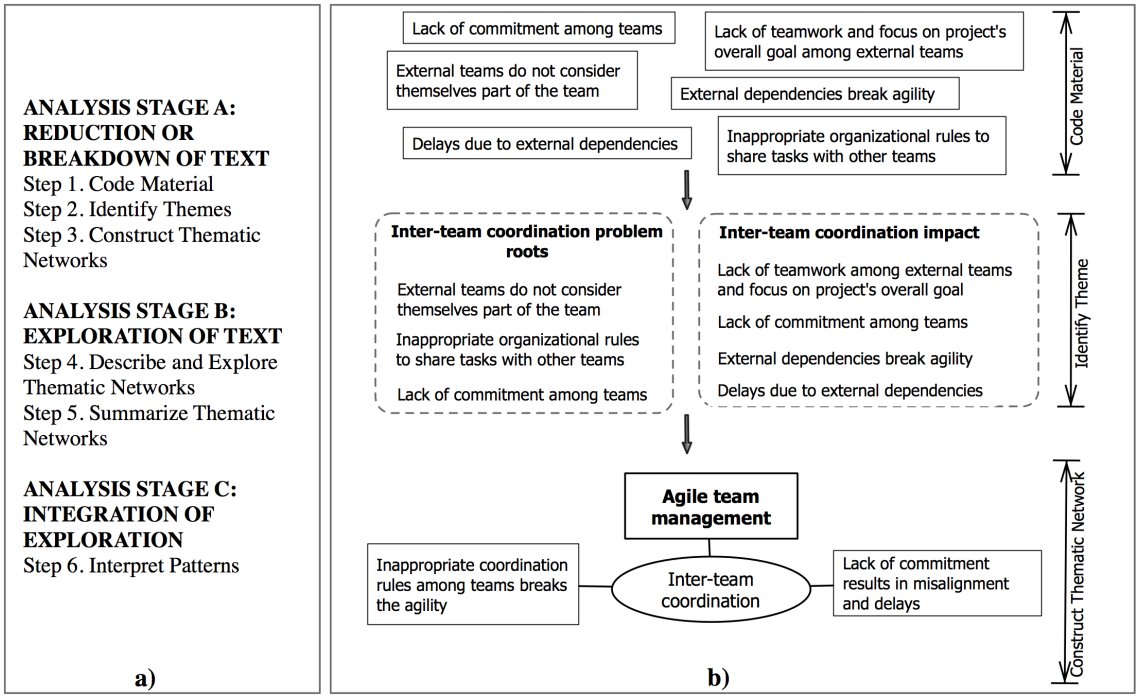

Figure 6.2: *a) 3-phase A-B-C qualitative method, starting with coding and ending up with thematic maps, b) example of Analysis Stages A and B*

data analysis, we used a tool, NVivo 9[1], which enables information classification into searchable codes.

Thematic analysis has limited interpretative power beyond mere description if it is not used within an existing conceptual framework [Braun and Clarke, 2006]. We thus adopted the conceptual framework on team productivity (Section 6.1) to anchor our analytic claims. Conceptual frameworks are useful as supports to better delineate qualitative studies and provide some clarity and focus; they can also be used to drive further discussion around the results [Miles and Huberman, 1994]. Other qualitative studies on agile team effectiveness [Moe *et al.*, 2010] and agile team communication [Pikkarainen *et al.*, 2008] have also presented conceptual frameworks as strategies to explain their results. It is important to note that we did not use the conceptual framework to guide the thematic analysis, but to discuss and report results. We thus use it to complement, extend, and verify our findings, an approach described by Corbin and Strauss [2007] for qualitative research.

To perform and thoroughly describe the thematic analysis, we used thematic networks that summarize the main themes constituting a piece of text [Attride-Stirling, 2001]. Thematic network

[1] Accessible at http://www.qsrinternational.com/products_previous-products_nvivo9.aspx

analysis contains three main sequential stages: *Stage A - Reduction or breakdown of the text*; *Stage B - Exploration of the text*; and *Stage C - Integration of the exploration*. While they all involve interpretation, a more abstract analysis level is accomplished at each stage. In this section, we describe all stages and the details of how our themes emerged. Figure 6.2.a summarizes these three stages and subsequent steps.

Stage A - Reduction or Breakdown of Text

Step 1 is to reduce the data, or *Code Material*. This may be performed by dissecting the text into manageable and meaningful text segments using a coding framework. This is a common procedure in qualitative research (Miles and Huberman [1994], Corbin and Strauss [1990]). This step in the analytic process is rather rudimentary, but it is imperative that it be completed with great rigor and attention to detail [Attride-Stirling, 2001].

Interviews transcription generated more than 400 pages. After transcribing the interviews, two researchers performed data coding, naming all possible productivity factors mentioned by the respondents. At this stage, 98 codes were generated. Two reviewers discussed each code before including it in the data collection tool (NVivo 9). After code generation, we reviewed each code in the raw information context. Appendix C presents evidence of code generation and analysis.

After all the text was coded, *Step 2* involved going through the text segments in each code (or group of related codes) and extracting the salient, common, or significant themes in the coded text segments. We had 12 themes after this step. We next went through the selected themes and refined them further into themes that are (i) specific enough to be discrete (nonrepetitive) and (ii) broad enough to encapsulate a set of ideas contained in numerous text segments.

Finally, the identified themes provided guidance for the thematic networks: *Team member turnover*, *Team design choices*, and *Inter-team coordination*. Figure 6.2.b illustrates the first stage of thematic analysis and the emergence of the *Inter-team coordination* organizing theme and its connection to the global theme found, *Agile team management*.

Stage B - Exploration of Text

In this stage, we returned to the original text, reading it linearly, theme by theme. The goal was to describe and explore the network (Step 4), supporting the description with text segments. Once a network had been described and explored, the next step was presenting a summary of the main themes and patterns characterizing it (Step 5). The objective here was to summarize the major themes that began to emerge in the network description and make explicit the patterns emerging in the exploration. Figure 6.3 describes the main themes and related patterns.

Stage C - Integration of Exploration

In this stage, the researcher brings together the deductions in the summaries of all networks and the relevant theory to explore the significant themes, concepts, patterns and structures that arose in the text (Step 6). The aim was to return to the original research questions and theoretical interests underpinning them and address them with arguments grounded on the patterns that emerged from exploring the texts. According to Attride-Stirling [2001], this is a complex and challenging task that is difficult to explain procedurally. We address this step in the Discussion (Section 6.4).

6.2.3 Data analysis for motivational factors

During the data analysis, we observed many notes and reflections from interviewees on motivation issues. Motivation did not appear as one of the most important factors in the thematic analysis, but it appeared together with important factors, such as team member turnover.

Motivation is one of the most frequently cited causes of software development project failure [DeMarco and Lister, 1999]. In Software Engineering, motivation is reported to have the single

largest impact on practitioner productivity and software quality management, and continues to be "undermined" and problematic to manage [Beecham *et al.*, 2008].

A motivated individual is one of the cornerstones of agile software development [Conboy *et al.*, 2011], explicitly mentioned as one of the twelve principles in the agile manifesto. This principle states: "Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done" [Beck *et al.*, 2001]. Therefore, managing and fostering motivation is critical for maintaining such agile methodology premise.

We decided to analyze our data to investigate possible connections between motivation and productivity, aiming at providing more empirical evidence on motivators in agile teams from industry. We needed some framework or guideline to support us to better identify and synthesize motivators. Our conceptual framework (Section 6.1) includes motivation as an individual characteristic of team members, but it fails to support the identification of motivators. We also wanted to make sure that our results were comparable with literature.

Sharp *et al.* [2009] recently presented a model to organize the knowledge about motivation in Software Engineering - the MOCC model (Motivators, Outcomes, Characteristics and Context). We adopted this model as a conceptual framework to identify and describe results. This analysis method differs from the previous one (productivity factors analysis) by using a deductive approach [Runeson *et al.*, 2012], in which the conceptual framework provides "insight, direction, and a useful list of initial concepts" [Corbin and Strauss, 2007]. In this case, the researcher remains open to new concepts and ideas to discover new concepts that did not appear in the framework. We thus performed thematic analysis from an initial list of concepts obtained from the MOCC model, discussing each code before including it in the tool.

6.3 Empirical results

Our objective is to provide a better understanding of some factors (and mediators) that impact agile team productivity. We describe, below, results from the multiple case studies analysis, presenting some quotations and other evidence that support the findings.

6.3.1 Key finding 5: The definition of Agile team productivity is diffuse

Based on the concepts of knowledge worker productivity (Section 3.1), we coded the answers in which the interviewees were talking about being more or less productive. We also coded any mention about their criteria to evaluate the productivity variation.

In most of the interviews, the team member's definition for productivity was unclear for the researchers. Three interviewees mentioned that *timeliness* is a criterion for measuring or perceiving productivity. Three interviewees also mentioned *quantity* and two mentioned *quality* as a way to determine productivity. It was surprising that only one interviewee mentioned *customer satisfaction* as a criterion, particularly if a team delivers high quality software on time, without satisfying the customer, they would not achieve the overall iteration or release goals.

6.3.2 Key finding 6: Agile team productivity factors are strongly related to team management

We describe below results from the thematic analysis, providing a short description for each theme and presenting quotations and other evidence that support the findings.

Figure 6.3 presents the results from the thematic network analysis of agile team productivity factors. In the following sections, we describe the organizing themes, *Team design choices*, *Team member turnover*, and *Inter-team coordination*, the main factors impacting agile team productivity, all related to the global theme *Agile team management*. The organizing themes have related roots and impacts on agile team productivity.

Team member turnover

Team member turnover occurred as a factor impacting team productivity. There was some degree of staff turnover in the three teams studied, as Table 6.1 shows. The teams perceived reduced productivity due to staff turnover.

Turnover can be defined as a type of membership change that involves the departure or arrival of a formally designated team member [Arrow and McGrath, 1995, Levine and Choi, 2004]. There are many categories of turnover expenses: separation costs (exit interview, administrative procedures); advertising and recruiting expenses; new employee orientation and training; and decreased productivity until the new employee is ready to contribute [Cascio, 1991, Tziner and Birati, 1996].

In Project A, at one specific time of the project, many team members left the project at once. Projects B and C experienced lower, but frequent, turnover. Developers said:

"There are things that impact the project negatively... One is the staff turnover. People who started the project are no longer here. Everybody now is new." (Developer, Project A).

"There was a drop in productivity at the end of the year when two developers left the project." (Project manager, Project B).

The turnover caused negative impact on team productivity and usually occurred due to job offers with better salaries or when a team member was unable to adjust to the team. Teams usually expect that newcomers adapt quickly to their fast-paced work. When this does not occur, teamwork might be compromised, affecting productivity:

"One of the problems of productivity today is the 'new' guy. This is a productivity issue. He has been here for about three months, but he didn't get into the rhythm of the team yet." (Developer, Project C).

In Project B, there was a tension between the QA (Quality Assurance) person and the other developers concerning work procedures. When the QA joined the team, he wanted to change some quality assurance procedures adopted by the team. In fact, the team was struggling with configuration management and testing tasks. The tension not only caused team members dissatisfaction, but also raised many conflicts in the meetings, which could not be completed on time. In the end, the company terminated the QA, which was considered both positive and negative by the team members.

In Project C, a developer joined the team but disagreed with some team procedures. The team did not accept the proposed changes, and the developer lost motivation. Here, we are not discussing the change merit, but the conflict origin. After a while, the developer left the company. In both cases, the newcomers tried to make changes in work procedures established by teams formed for at least six months (Section 6.2.1) and faced barriers that led to the turnover.

Conversely, team members also mentioned a positive staff turnover influence on their productivity: the opportunity for the team to improve and grow. New team members can bring new ideas and experiences, leading the team to a more mature level. This relates both to the team's ability to handle turbulent environments and the continuous learning ability.

"We see new people's arrival on the team in a different perspective. Maybe their proposals seem to be awkward for the team and they [the team] need to mature to a point in which maybe the proposed ideas will be good." (Developer from Project B).

In the retrospectives of the three companies, we found evidence supporting the positive side of turnover: new people bring more energy to the group, especially when the team motivation was deficient. However, the results were stronger in Projects A and B, where the turnover was medium but somewhat frequent. In Project A, the turnover was also medium but happened once a year.

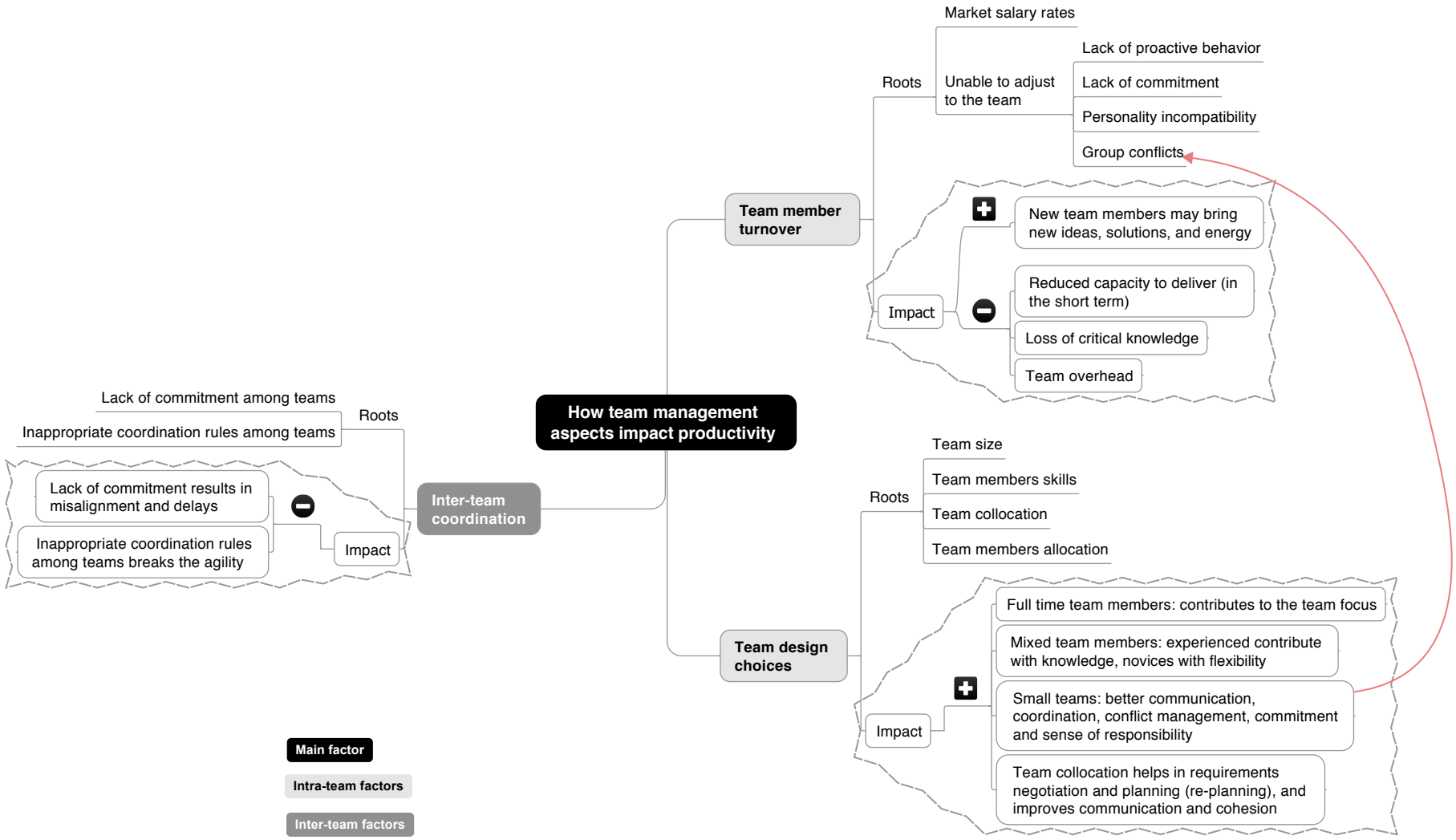

Figure 6.3: *Thematic map on agile productivity factors*

There are many occurrences of positive notes in the subsequent retrospectives after team member turnover. In a retrospective session, teams explicitly divide negative and positive aspects of the previous iteration to recognize positive actions and discuss improvements for the negative ones. We identified positive notes greeting new members, sometimes referring to *"new blood in the team"* - a Brazilian expression that connotes a feeling of renewed energy.

Team design choices

We found that *Team design choice* is a factor impacting agile team productivity. Team design choices are the member attribute configurations in a team [Levine and Moreland, 1990]. Our findings indicate desirable team design attributes: full-time allocation, diversity (mixed teams), team member skills, team size, and collocation (physical proximity). Team composition with different profiles and knowledge levels was considered positive for team productivity, especially in Project A.

Considering the organizational context and business, Project A clearly contains high-ability (business experts, experienced software architects) and low-ability workers (novices). One possible explanation is that the benefits of having mixed team members on such teams may be more noticeable than for the other teams. Experienced team members contributed to the work by adding knowledge, while the others contributed by being flexible, as stated by a developer:

"Some things contribute to productivity. We have very experienced people here and less experienced ones that are more flexible" (Developer, Project A).

The respondents mentioned that small teams lead to better communication and alignment. In addition, conflict management and coordination among team members are easier to handle. As team size increases, the number of necessary communication links between team members increases and there will be more potential conflicts to manage. In Project B, some members left the project, and the remaining team members were allocated full-time to the project. A Developer said the following:

"As we reduced the team, we are starting to focus more on what we want. Before, it was a bit messy and people did not perform certain tasks because they thought that other people would do it" (Developer, Project B).

In the same project, the product owner also noticed the benefits of the full-time allocation:

"After the reorganization, nobody needs to move to other activities outside the project. So, they are more focused. Everyone knows everything in the project." (Product Owner, Project B).

In Project A, the team size increased over time, and some team members noted it as a negative factor impacting team productivity. From the team members' perspective, small teams also enable a better understanding of the product's big picture because fewer people need to learn and keep up-to-date on the product scope. In addition, respondents said that small teams help increase the team's sense of responsibility and commitment.

Our respondents also mentioned team collocation as an attribute. This result was stronger for Project A, considering the frequency of references during the interviews. Team members in this project explained that collocated work helps overcome the invisible barriers between teams in a hierarchical company. They noticed improvements in requirement negotiations and risk mitigations through the socializing atmosphere provided by collocation.

Respondents mentioned that their productivity depends on their workspace layout. Both projects were radically collocated [Teasley *et al.*, 2000], but they mentioned that it is not enough to be in the same space. The layout (including desk positions and proximity) may also impact their productivity.

Inter-team coordination

We found that *Inter-team coordination* impacts agile team productivity (Figure 6.3). There are several kinds of dependencies in a project. Shared resources, prerequisite constraints, simultaneity constraints, and the relationship between tasks and subtasks are common examples of dependencies among activities in a project [Malone *et al.*, 1993]. Coordination processes are commonly used for managing dependencies among activities [Malone and Crowston, 1994, Malone *et al.*, 1993], enabling teamwork among different teams.

In large organizations, software development teams often depend on other teams to accomplish their tasks. These include external customers not collocated in the project; operation teams helping publish versions of the system or data models across different environments (integration, homologation, production); external QA teams verifying compliance between the developed system and organizational rules; and other development teams providing reusable assets to the project.

Through the interviews and retrospectives, we have identified evidence that external dependency management represents a recurrent problem in the three studied organizations. Our direct observation field notes corroborate this factor and there were references to this during the daily meetings and plenty of tasks waiting for impediment resolution resulting from external dependencies.

Companies coordinate dependencies among teams or resources using certain strategies. When Project A delivers the system to the test environment, it must submit some artifacts, such as data models, to the QA team. The QA team is an example of a resource shared among all enterprise projects in the company. It implements a "first come/first served" coordination process [Malone *et al.*, 1993] to manage requests from other teams. According to the team, this kind of coordination solution is misaligned with the pace of the agile project:

"The other teams are not working at the pace of the project, they are not working in the (same) way... The organization is not ready yet" (Developer, Project A).

A similar problem occurs when Project B publishes the system to the corporate environment:

"Currently, there is one factor that we are dealing with after a lot of feedback from our retrospective, which is about the relationship among the boundaries of development, testing, and production. Whenever we cross these boundaries, a bottleneck occurs." (Project Manager, Project B).

Another external dependency problem occurs when an agile team decides to reuse components or existing systems, building a system of systems. External components and systems are often evolving and have their own lifecycle. Agile teams must sometimes wait for some components, which may compromise a timely delivery, the "unsynchronized agile release pattern" [Leffingwell, 2007]. Figure 6.4 illustrates the lack of synchronization among teams that are working on the same project. The main team (Team A) requests components or services from external teams, such as Teams B or C. While waiting for the requests to be completed, the main team thinks it is on track. However, when the planned system release date arrives, the team slips into integration problems.

Project A reuses several components and must implement interfaces to communicate with other systems. The problem with the integration with components and systems is thus:

"The productivity at the end of the first phase of the project will be somewhat lower because we will have solved a lot of problems... which are the integration with other systems, publishing to a server with other business components." (Project Manager, Project A).

Project C also faces the same inter-team coordination problem, even when the other team uses agile methods:

"Sometimes we have integration with other systems, internal but complicated ones. You're committed, you're on deadline to deliver, but the other team is not; or the other (team) can even be

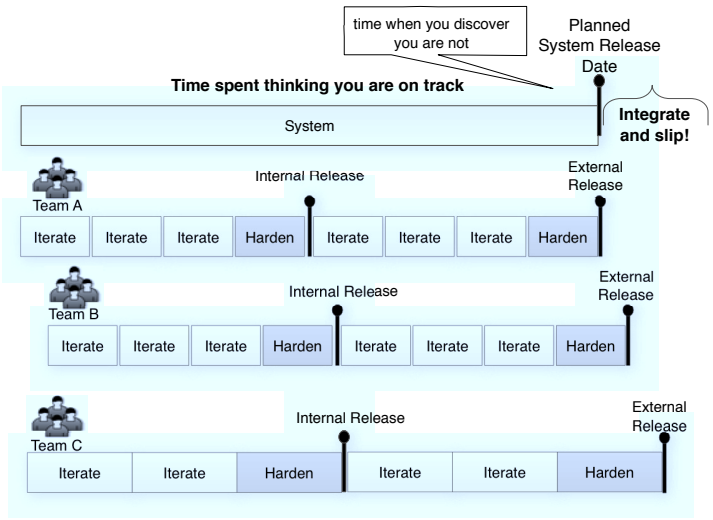

Figure 6.4: *Unsynchronized agile release pattern (adapted from [Leffingwell, 2007])*

committed, but they are not able to respond on time. They are also working in some release, and probably will tell you: 'oh, to make such a change, just in the next sprint' " (QA, Project C).

These solutions for managing dependencies are thus not compatible with the team needs. For agile projects, it is not only important to manage dependencies, but also to resolve them in a timely manner. Thus, agile teams must be more synchronized to achieve this final goal [Leffingwell, 2007].

Respondents also mentioned that external teams sometimes are not committed to the project goal, only with the execution of the requested task. In general, they do not consider themselves part of the team and tend to overemphasize their importance in the process. A Developer comments:

"There are a lot of roles, such as 'I am the system administrator', 'I am the QA.' They don't understand that we're a multidisciplinary team. Reducing the conflict between these roles can simplify our work process" (Developer, Project B).

The emphasis on their specific roles denotes not only a lack of teamwork spirit, but also an attempt to use their roles as a means to enforce the choice of practices to use by each team member. A system administrator would thus use his/her "role" to determine how the team should proceed in administrative matters. This seems to cause more trouble than potential benefits in the inter-team coordination processes definition.

6.3.3 Key finding 7: Pair programming and Collocation as key practices influencing team productivity

In order to answer SQ3, we coded all answers that relate an agile practice to team productivity. We then ranked the most cited practices and selected only those mentioned at least by 30% of interviewees. Table 6.4 summarizes the two agile practices related to team productivity, a brief description of the practices, the percentage of responses used to rank them, the impact on team productivity, and the perceived effects of the factor.

Pair programming

Pair programming is the XP core practice and involves two programmers working collaboratively to develop software. Dybå *et al.* [2007] conducted a meta-analysis on 18 studies regarding pair and solo programming. As a result, the authors provide guidelines for when to use pair programming based on the programmer expertise and the task complexity. They recommend that Intermediate

Table 6.4: *Most perceived agile practices, definitions, and effects on productivity*

Agile practice	Description	% / Number of responses	Impact	Effects
Pair programming	Pair programming is an XP core practice and involves two programmers working collaboratively to develop software [Balijepally *et al.*, 2009]	53% (7 out 13) (4 from Project A, 3 from Project B)	Positive and Negative	(+) Enables knowledge dissemination o Even when team members work on different schedules. o Especially when people have different backgrounds (e.g., domain vs technical). (+) Helps in creating commitment, supportiveness and trust. (+) Increases the level of communication. (-) Full time pair programming o Counter-productive and demotivating in easy tasks o Controversial issue when resolving complex tasks - it depends on the individual cognitive process of problem solving.
Collocation	Consists of placing team members near each other.	30% (4 out 13) (2 from Project A, 2 from Project B)	Positive	(+) Helps in requirements negotiation and planning (re-planning) (+) Improves communication and cooperation. (+) Varies as a function of the workspace layout o Even when the team is collocated, team members want to be "radically collocated"

and Senior developers should not work in pairs to solve easy tasks, because it could be counter-productive. Our results show that some team members also consider using pair programming to solve easy tasks a waste of time.

"Despite being an advantage to use pair programming, when some activities are very simple, we do them individually. I mean, if you always work in pairs, I think there are times when you will lose productivity." (Developer, Project A)

"I think that pair programming brings a lot of benefit, works very well, but should not be used 100% of the time. Depending on the task, when it is very basic, repetitive, two guys working on that is unnecessary." (Developer, Project B)

On the other hand, pair programming is also seen to contribute to increased satisfaction, keeping team members motivated [Balijepally *et al.*, 2009, Williams *et al.*, 2000]. The results from our interviews were, however, different when interviewing intermediate and senior developers. Developers mentioned feeling demotivated when using pair programming all the time, because, sometimes, it is tiring or the task is simply too easy to be done in pairs. They also mentioned that, when solving complex tasks, they feel demotivated to work in pairs because they would like to have time to think

about the problem alone before discussing it.

"I think that pair programming is productive, but I don't like to work in pairs... because I'm introspective. Sometimes, I like to be alone and start thinking, without much talking, solving the problem there, developing my solution." (Developer, Project A)

"Sometimes pair programming doesn't help. When you don't know in which direction you should go to solve complex problems. You want to use your own usual process to test things, trial and error, random exploration. This is not easy to explain to others." (Developer, Project A)

Collocation

The collocation of teams is an approach aiming to improve communication and collaboration among team members. Both Scrum and XP recommend collocation as an agile practice. When adopting this practice, companies usually hope for productivity enhancement [Teasley *et al.*, 2000], but there are advantages and disadvantages with the use of collocation in software development [Eccles *et al.*, 2010, Hinds and Kiesler, 2002, Rafii, 1995, Teasley *et al.*, 2000]. In our study, some projects reported significant productivity gains, and improvements in communication, spirit of teamwork, learning and motivation. Lack of privacy, work interruptions, lack of individual recognition, and some disconnection from the rest of the teams were mentioned as the negative side of collocation.

Collocation was mentioned by 30% of our interviewees. This result was stronger for Project A. Team members in this project explained that collocated work helps to overcome the invisible existing barriers between teams in a hierarchical company. They noticed improvements in requirement negotiations and risk mitigations through the socializing atmosphere provided by the collocation.

The interviewees mentioned that their productivity varies according to the workspace layout. Both projects work radically collocated, i.e., work in the same physical space [Teasley *et al.*, 2000], but they mention that is not enough to be in the same space. The layout (such as the desks positions and proximity) may also impact on their productivity. This factor was mentioned more by interviewees in Company A than in Company B. This may be because Company B has invested in a customized layout that benefits working in pairs, while Company A has kept the same traditional workspace infrastructure.

6.3.4 Key finding 8: New motivators might influence agile team productivity

We identified 33 general motivators in the three agile companies. We organized them into patterns and then classified using the MOCC model [Sharp *et al.*, 2009], seeking to standardize the description of each motivator using a common language. Finally, 11 motivators emerged from the data as general motivators in agile teams, as shown in Table 6.6. We ranked the motivators by their relative frequency in the results.

The most frequent general motivator we found is technically challenging work (M1), in which work is not mundane and is technically interesting [Hall *et al.*, 2008]. Despite being the most frequent motivator, it is definitively stronger in Company A. It is hard to precisely define the reasons for this specific result. However, considering the company context, there are some clues. First, the project in Company A involves many other financial institutions and it is the first agile project adopting the technology of automated acceptance tests instead of conventional document-based requirements specification. The project is a priority for the company and the system has to interact with many legacy systems. This challenge probably provided special motivation for team members. Another consideration is that many team members we interviewed in Company A are new in the Company, or at least new in the business unit, which may provide more technical challenges. This also might explain why Problem solving (M11) appeared only in Company A.

Team working (M2), *Identify with the task* (M3), *Employee participation/involvement/working with others* (M4), and *Development needs addressed*(M6) are motivators mentioned at least thrice.

Except *team working* (M2) and *Development practices* (M10), all of them are motivators not inherent to Software Engineering. They are intrinsic motivators related more generally to the work. Just one of them, *Good work/life balance* (M9), is an extrinsic motivator. Therefore, it seems that intrinsic motivators not necessarily related to Software Engineering are the most important motivators on the studied agile teams.

Only one motivator appeared in the three companies: *Development needs addressed* (M6). It is defined as existence of training opportunities to widen skills and opportunity to specialize [Sharp *et al.*, 2009]. The interviewees stated knowledge sharing and high level discussions as the concrete path for specialization and training. Agile development promotes intensive communication as a way to foster knowledge sharing and team building. Another issue mentioned was the company's know how as a good way to specialize. This is a company's intangible asset or intellectual capital that goes beyond the agile team.

During the motivators classification, we decided to describe a new motivator *Working in a successful product* (M5), not present in the MOCC model. In fact, the model describes a similar motivator named as "working in a successful company" [Sharp *et al.*, 2009]. In our results, interviewees were not only proud to work in a successful product, but also demonstrated a strong commitment to the product performance in the market. If the product goes well, team members feel motivated. If a rival product threatened their product in the market, they behave competitively. Thus, we noticed an impressive alignment between agile team members and companies' business goals, which give them a new motivator. This result appeared in Companies B and C only, which is somewhat expected, since Company A was not developing a product.

We noticed similarities between Companies A and C regarding motivators possibly related to the companies' organizational structure. *Employee participation / involvement / working with others* (M4), *Autonomy* (M7), and *Equity* (M8) emerged as motivators just in these two primarily vertical, hierarchical companies. These motivators seem to be promoted because of the agile team arrangement, since vertical structures are well-known inhibitors of strong involvement between employees, also limiting equity and autonomy [Galbraith, 1973]. Both companies were previously organized to segregate teams by their specialization. Now, the employees are working in whole teams [Beck and Andres, 2004], i.e., agile teams with all skills and functionalities needed for creating the product (developers, testers, designers, technical writers, and customers). In contrast to their previous experience, this new organization seems to be a great and noticeable motivator. Those motivators did not appear in Company B, whose structure is horizontal and which embraces agile methods longer than the others.

Finally, one development practice – whole team – was mentioned as a motivator in Companies A and B. The practice emerged without any specific question regarding agile methods, it appears in the context of general motivators. We believe that, in Company A, this motivator is related to the company structure (as mentioned before). Company B had just hired specialists in user experience (or user interface/UI), solving existing problems in the project. Thus, the team feels comfortable to deliver better software, because they had the needed capability.

6.4 Discussion

Through an interpretative field study in three large companies in Brazil, we investigated factors that affect agile team productivity. We now discuss the cases in light of our sub-research questions:

- *SQ1. How do agile teams define productivity?*
- *SQ2: What do agile team members identify as the main factors impacting on productivity?*
  - *How do these factors impact positively, or negatively on the productivity of agile teams?*
- *SQ3: Which agile practices are perceived to impact on a given team's productivity?*

Table 6.5: *Cross-case analysis of General Motivators in Agile teams*

	General Motivators	Company A	Company B	Company C
M1	Technically challenging work	****	*	
M2	Team working	*	**	
M3	Identify with the task (clear goals, personal interest, know purpose of task, how it fits in with whole, job satisfaction; producing identifiable piece of quality work)	***		
M4	Employee participation/involvement/working with others	**		*
M5	Working in a sucessful product		*	**
M6	Development needs addressed (e.g. training opportunities to widen skills; opportunity)	*	*	*
M7	Autonomy	*		*
M8	Equity	*		*
M9	Good Work/life balance (flexibility in work times, caring manager/employer, work location)	*		*
M10	Development practices (object oriented, XP and prototyping practices)	*	*	
M11	Problem solving (the process of understanding and solving a problem in programming terms)	**		

Our preliminary results show that not only is productivity a diffuse concept but also, especially in the agile teams studied, it remains very much associated with *quantity*. In other words, for many of the interviewees, team productivity is the ratio of outputs to inputs. The story points delivered are the output and the iteration is the unit for the input, i.e., it represents the time spent to deliver the story points. Similarly, XP proposes project velocity as a productivity metric for team performance [Beck and Andres, 2004]. The drawback of this metric is that many teams use it as a way to show that the team is able to move rapidly, but it fails to show whether the teams are going in the right direction. The concepts of *timeliness*, *quality*, and *customer satisfaction* were **not** strongly associated with productivity by our interviewees, which contradicts the priorities of an agile team posed by the Agile Manifesto.

The productivity of the studied agile teams was more sensitive to *team management issues*. Agile teams often have responsibility for managing their own work and behaviors; others usually make decisions about goals, team structure, and organizational supports [McHugh *et al.*, 2011]. However, often these 'other' choices influence the team, which in turn requires attention from team members. Our analysis indicated *team design choices* and *staff turnover* as intra-team management factors impacting productivity and *inter-team coordination* as an inter-team management factor. Furthermore, we identified some relationships between pair programming and collocation practices and agile team productivity. In this section, we discuss our major findings.

6.4.1 Intra-team management factors and a revised conceptual framework

We refined the conceptual framework (Figure 6.1, Section 6.1), incorporating our results. The framework was developed based on a review of teamwork effectiveness literature in traditional and agile teams. In this section, we integrate the organization experiences with factors impacting agile team productivity. Figures 6.6, 6.5, and 6.7 depict our findings as relationships among input, group

Table 6.6: *Cross-case analysis of Specific Agile Development Motivators in Agile teams*

	Agile Motivators	Company A	Company B	Company C
AM1	<NEW> Feeling of progress/accomplishment	****	*	***
AM2	Development needs addressed (e.g. training opportunities to widen skills; opportunity)	**		**
AM3	<NEW> Lack of bureaucracy in the development process	**		*
AM4	Technically challenging work	*		*
AM5	Good management (senior management support, team-building, good communication)	**		
AM6	Feedback (from the job, from supervisors, on goal accomplishment)	**		
AM7	Employee participation/involvement/working with others	**		
AM8	Experiment (trying something new, experimentation in order to gain experience)			**
AM9	<NEW> Elimination of waste (e.g., Automated acceptance tests as executable requirements, don't have to execute tests manually all the time)	**		

processes, and outcomes. In the revised framework, we detail how the team management aspects – staff turnover, team design choices, and inter-team coordination – provoke changes in agile team productivity.

We revised the framework using the following process. For each theme, we analyzed its roots and impact on productivity as well as the existing links on the original conceptual framework. When the link already existed, we marked the impact (positive/negative). Otherwise, we created the link between input and outcomes, also considering possible group processes mediating the relationship. Small teams led to better communication, easier conflict management and coordination, and, ultimately, agile team productivity. We thus created, in Figure 6.5, a link between the input *Small teams* and related group processes, including *conflict management*, *coordination*, and *communication*. Afterwards, we assigned a link between these processes and the team productivity outcome because they appear in the context of factors impacting agile team productivity. Finally, we marked the type of the impact – in this case, positive.

Team member turnover. The initial conceptual framework proposed staff turnover as an input factor that impacts team productivity, mediated or not by some group processes. In our findings, turnover is an input and outcome, serving as input back into the team processes[2].

Staff turnover is a common team trouble spot for any software development project [Abdel-Hamid and Madnick, 1991, Wallace *et al.*, 2004]. Coram and Bohner [2005] state that high turnover in an agile team can lead to losing critical knowledge due to the lack of documentation. The turnover may happen for reasons originating within the group, such as personal disagreements, or external variables, including retirement or job opportunities outside the company. The group must adapt to turnover, despite the reasons for it.

[2]In fact, turnover has occupied both an input role and an output role in traditional IPO models of teamwork effectiveness [van der Vegt *et al.*, 2010]

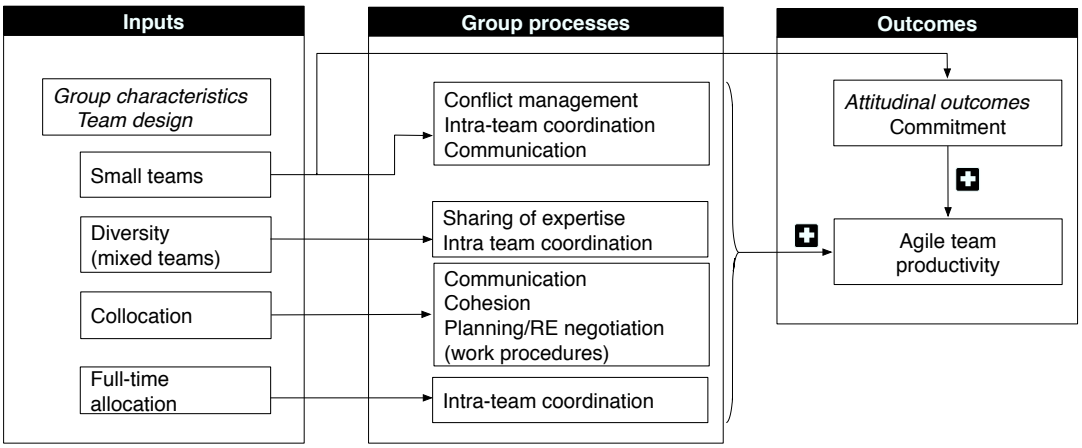

Figure 6.5: *Team design choices factors and effects on agile team productivity*

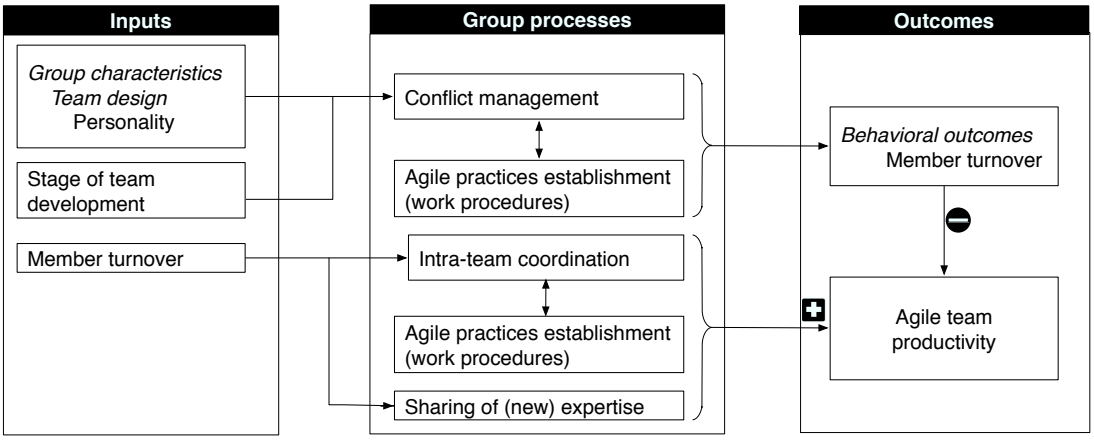

Figure 6.6: *Staff turnover factors and effects on agile team productivity*

Most empirical research on turnover analyzes its impact on team productivity (and other group performance indicators) without clarifying possible mediation processes [Levine and Choi, 2004, van der Vegt *et al.*, 2010]. Knowledge regarding the specific group processes that might intervene with these indicators is therefore limited [Brian R Dineen, 2003, Kacmar *et al.*, 2006].

Our results show a negative impact of turnover on agile team productivity. We observed (and the team members reported) many conflict episodes regarding agile work procedures, especially quality and configuration management. Our analysis suggests that *personality* and the *stage of team development*, mediated by the selected *conflict management processes* may result in *member turnover* and impact on *agile team productivity*. In general, teams in the storming stage [Tuckman, 1965] face conflicts, reconcile, and evolve by setting new norms. When the conflict management does not succeed, teams expect to observe commitment loss, leading to dismissals and turnover. This happened in the observed teams. On some level, they were not able to manage the conflicts regarding the work procedures, leading to turnover and decreased productivity in the short term (teams were unable to deliver for a while), as well as loss in both knowledge and team overhead after the turnover.

Our findings on the influence of *personality* in the conflicts arising in team work (Figure 6.6) are consistent with Licorish *et al.* [2009], who observed that the greater diversity of individuals involved in agile teams, combined with the less rigid nature of their involvement, may increase the incidence of personnel incompatibilities and, therefore, the potential for conflict. Hoda *et al.* [2010] found that some team members may not have the desired attributes to be part of an agile team

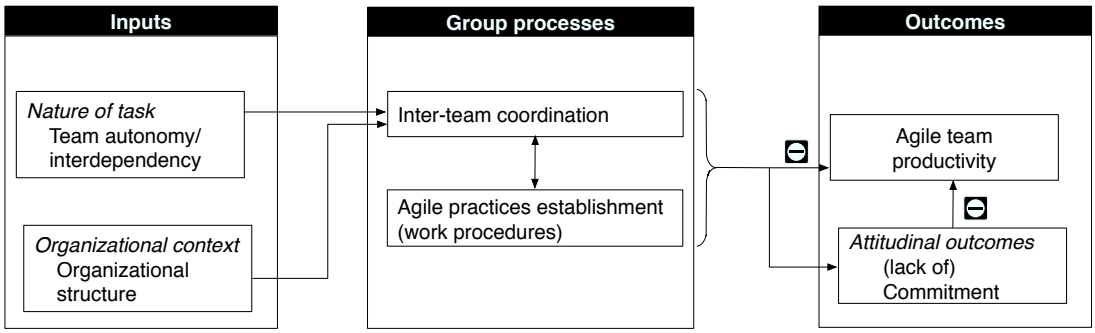

Figure 6.7: *Inter-team coordination factors and effects on agile team productivity*

and are perceived to pose a threat to the proper functioning and productivity of a self-organizing agile team. *Personality* and individual practices were seen as root causes that led to turnover. This is somewhat consistent with our results, as *personality* was as input that influenced member turnover, which implies decreasing productivity. However, Hoda *et al.* [2010] assume that the team was doing well and an individual caused malfunction. Our results, in contrast, suggest that both the team development stage and conflict management processes matter and interact with personality, resulting in higher team member turnover.

Although agile methods embrace conflict and dialectics [Nerur and Balijepally, 2007], our results show that there is a limit of tension and conflicts that teams can tolerate. Empirical evidence from teamwork literature has supported the negative relationship between conflict and team productivity because it produces tension, antagonism, and distracts team members from performing their tasks [Dreu and Weingart, 2003]. Conversely, according to McAvoy and Butler [2007], it is possible to maintain cohesion while introducing low levels of positive conflict in an XP team, which will guarantee better decision-making and learning outcomes. In such self-organized teams, some level of process conflict seems inevitable because there is no legitimate authority to enforce process rules or prevent process conflicts (disagreements about assignments of duties and resources) [Behfar *et al.*, 2011]. However, it is still not clear which degrees (and types) of conflict are healthy for agile teams. Our results suggest that process conflicts, including how to accomplish and divide work, may decrease agile team productivity by causing team member turnover.

Member turnover also plays an input role in our results (Figure 6.6), as it generates other indirect effects on agile team productivity. The agile teams had to self-adapt and reorganize their routines, which took time and negatively impacted team productivity. Conversely, new team members may bring new ideas, solutions, and energy to *establish agile practices* and make improvements in the *team coordination processes*. Those changes positively impact agile team productivity. Our results confirm previous research on teamwork (not specifically on software development) [van der Vegt *et al.*, 2010] that acknowledges high turnover as a potential disrupter of routines, norms, group composition, and a way to introduce new ideas, all of which have important implications for team effectiveness and team productivity.

Team design choices appeared to contribute a great deal to agile team productivity, corroborating previous research on teamwork [Chow and Cao, 2008, Cohen and Bailey, 1997, Maxwell and Forselius, 2000, Tan *et al.*, 2009, Yeatts and Hyten, 1998]. Figure 6.5 depicts our findings in light of the conceptual framework presented in Figure 6.1, Section 6.1.

First, the respondents mentioned that *small teams* led to better *communication*, alignment, and *commitment*. In addition, *conflict management* and *intra-team coordination* were easier to handle. Such responses confirm findings from other studies [Blackburn *et al.*, 2000, 1996, Brooks, 1995]. As team size increases, the number of necessary communication links between team members increases, leading to more potential conflicts to manage. Having a small agile team thus leads to higher productivity. We also observed a relationship between the *small teams* and smaller turnover, probably due to a reduction of intra-group conflict. This result emerged in our thematic map (Figure

6.3) and is better explained through the revised conceptual framework in Figure 6.5.

Second, team *diversity* (mixed teams) emerged as a factor contributing to agile team productivity. Our results showed that agile teams with a mix of experienced and non-experienced workers may have a positive impact on productivity due to the knowledge and flexibility they can provide, respectively. This result suggests that there are benefits in maintaining agile teams with diversity in experience. Our results are consistent with previous research on manufacturing teams [Hamilton *et al.*, 2003], where workers may have both technical and collaborative skills (such as flexibility) to be more productive. In the same direction, Lee and Xia [2010] found that diversity is an important team variable to build team software development agility; however, they did not report the relationship between knowledge and flexibility in mixed teams.

Third, team *collocation* intends to improve communication and collaboration among team members. Both Scrum and XP recommend collocation as an agile practice. Adopting such a practice, companies also hope for productivity enhancement [Teasley *et al.*, 2000], but there are advantages and disadvantages in using collocation in software development [Eccles *et al.*, 2010, Hannay and Benestad, 2010, Hinds and Kiesler, 2002, Rafii, 1995, Teasley *et al.*, 2000]. Our results confirm previous research, as the projects reported significant productivity gains through improvements in *communication*, teamwork spirit (*cohesion*), *planning*, and requirements *negotiation* when collocated. Lack of privacy, work interruptions, lack of individual recognition, and some disconnection from the rest of the teams were mentioned as the negative side of collocation.

Finally, *full-time allocation* of team members was considered positive for overall agile team productivity. It enhances team focus to complete tasks, decreases work interruptions and distractions, and increases team member awareness of the project situation. When team members were allocated part-time to the projects, they were not able to follow the project status well, even by participating in daily meetings. Our results complement previous research on agile team effectiveness [Moe *et al.*, 2009], showing that developers working on two or more projects in parallel had to manage conflicts among different team goals or needs, damaging a self-managed team's potential.

6.4.2 Inter-team management factors and a revised conceptual framework

Recent research has been devoted to understanding agile intra-team coordination [Sharp and Robinson, 2010, Strode *et al.*, 2011] and its relationship with agile team effectiveness and productivity [Mishra and Mishra, 2009, Moe *et al.*, 2010]. However, agile teams are often embedded in large organizations, dealing with other teams to accomplish their goals. In large companies, such as the ones we studied, it is common to find agile projects with several teams, many of them sharing various projects. Despite initial success at the team level, some teams then find it difficult, if not impossible, to implement agile methods beyond their own boundaries [Abrahamsson *et al.*, 2009].

Coordination between software development teams is one of the most difficult-to-improve aspects of software engineering; its importance increases as software development becomes distributed [Begel *et al.*, 2009]. In fact, teams do not function "in a vacuum", and all external activities (so-called boundary activities) may influence team performance and effectiveness [Joshi *et al.*, 2009]. Figure 6.7 depicts our findings in light of the conceptual framework presented in Figure 6.1, Section 6.1.

Our findings indicated that *team interdependencies*, mediated by *inter-team coordination* processes, result in decreased agile team productivity. The interdependencies range from QAs, Operations, and External customers to maintenance and other project teams. The coordination strategies seem to be wrong, especially for handling the agile team pace and priorities. Using queues to request tasks between agile teams is not a good coordination strategy because it does not handle synchronization properly. Moreover, it was difficult to establish priorities in a timely manner between two agile teams due to the (natural) presence of uncertainty in their projects. Inappropriate coordination rules break team agility, resulting on delays and not achievement of the iterations goals.

The negative impact was more notable in Company A, whose organizational structure is more rigid and coordination processes between units are primarily vertical (via supervisors, line managers, or other hierarchy representatives). In Companies B and C, we observed both vertical and

horizontal coordination. vertical coordination are usually implemented through project managers, while linkage in horizontal coordination is provided by an individual team member who communicates directly with other members or users on a one-to-one basis [Galbraith, 1973, Parolia *et al.*, 2007]. The organizational structure thus seems to accentuate the negative impact of some inter-team coordination processes on agile team productivity. However, the intensity of this relationship should be further explored.

Although agile teams follow the organizational procedures to send requests to other teams, they perceive that the other teams are not really committed to their projects, which impacts agile team productivity. Our interpretation, based on all data collected and observations, is that the adopted inter-team coordination procedures do not favor establishing common goals among teams, which leads to the observed lack of commitment. Our results confirm previous research on team performance, where coordination process choices influence commitment and clear mission establishment, which, in turn, impact team performance [Parolia *et al.*, 2007]. Our results also shed light on the research topic suggested by Abrahamsson *et al.* [2009] regarding synchronization practices of agile and non-agile functions.

6.4.3 Pair programming, collocation and agile team productivity

One striking result from our study was the connection of pair programming with tasks and motivation. When tasks are too easy or too complex, they may influence the motivation to work in pairs. As we have not found references to this phenomenon in the literature, we believe this is a point for further investigation in the use of pair programming for agile team productivity. In addition, collocation of teams seems not to be enough to reach high productivity in agile teams, but it seems that there is a need to invest in the workspace layout.

The following hypothesis can be derived from our results: i) the degree of complexity of the task affects motivation to work in pairs, which in turn, affects the productivity; ii) productivity varies in function of the workspace layout, even when teams are radically collocated. There is a need for further investigation of these hypotheses in new empirical studies.

6.4.4 Team members motivation influencing agile team productivity

We found 11 general motivators in the case studies. 5 motivators confirm literature [Melo *et al.*, 2012]: Team working (M2), Identify with the task (M3), Development needs addressed (M6), Autonomy (M7), and Problem Solving (M11). These motivators, therefore, are strong in agile teams. We also found 9 agile motivators, 3 of them confirmed the literature evidence [Melo *et al.*, 2012]: Development needs addressed (AM2), Good management (AM5), and Feedback (AM6). The strongest result, therefore, is that agile methods address *development needs of specialization and training* (M6/AM2), since we found it twice in our case, and also in the review.

Seven motivators emerged only in our case studies; we did not find them in literature [Beecham *et al.*, 2008, Melo *et al.*, 2012, Sharp *et al.*, 2009]. They are: Technically challenging work (M1), Employee participation /involvement/working with others (M4), Equity (M8), Development practices (M10), Technically challenging work (AM4), Employee participation / involvement / working with others (AM7), and Experiment (AM8). Our results suggest that those motivators are also important for agile teams and should be further investigated.

In addition, we found four new motivators in our case studies: Working in a product that is successful (M5), Feeling of progress/accomplishment (AM1), Lack of bureaucracy in the development process (AM3), and Elimination of waste (AM9). They denote that agile development not only helps on a better alignment between IT and business (M5), but it increases the sense of contribution (AM1) and supports process improvement (AM3 and AM5). All of them motivate team members. It is important to note that waste is the term used by team members during the interviews. It is not necessarily related to the adoption of Lean development.

We found that agile teams have Good Work/life balance (M9), considered as a motivator. Conversely, McHugh *et al.* [2011] found that teams feel stressed due to the frenetic pace of agile

work, having to always deliver every day and sometimes working overtime when the pressure is very intense. Thus, there are variables influencing teams to consider or not agile methods supports life balance. Comparing our companies' profiles to McHugh's companies, the major difference that may explain the opposite results is the length of time since agile implementation. Our teams are, at least, 2-year experienced on agile, while McHugh's are between 9 months and 2 years. Thus, experience in using agile might bring the balance needed to foster motivation, thus productivity. However, this should be better investigated in future studies.

6.4.5 Implications for theory and practice

In agile software development, few studies discuss the implications of staff turnover, considering both intervening processes and outcomes. For instance, conflict types and conflict management processes and their impact on group performance are issues that have received little research attention [Behfar *et al.*, 2011]. Our findings shed light on inputs, such as *personality* and *stage of team development*, and group processes, such as *conflict management* and *agile practices establishment*, which result in turnover in agile teams. We also offer possible explanations on the positive impact of turnover on agile team productivity, mediated by group processes, including *intra-team coordination*, *work procedures establishment*, and *sharing of new expertise*.

Team design choices remain an important factor impacting team productivity, and it is even more pronounced on agile teams that rely on teamwork and people factors. More research on tool support for agile team composition, such as that conducted by Licorish *et al.* [2009], is needed. Our results indicated that *team size*, *diversity*, *personality*, *skills*, *collocation*, and *time allocation* are key aspects to be considered when designing agile teams.

Our findings suggest that companies may need to review their own structure and determine the fit between their structure and agile teams. Company structure decisions directly influence the coordination process selection among teams. Achieving alignment between teams is challenging and, in our view, poses a problem for both corporate-level and team management. Inter-team coordination emerged as a productivity factor in agile teams, but the teams themselves cannot change organizational processes. The *intra-team coordination processes* processes must be adjusted to enable productive work by considering priorities and pace between teams. Based on our preliminary findings regarding pace and priority issues, there is an avenue for further research on the influence of inter-team coordination strategies and structure on agile team productivity.

Because agile methods are people- and team-oriented [Abbas *et al.*, 2008], establishing teamwork and managing people are crucial to deploy agile methods effectively. The most important implication to managers working with agile methods is that it places more emphasis on people factors in the project, so the "attention to the human issues gives agile projects a particular feel" [Cockburn and Highsmith, 2001]. People factors directly influence the ability to work in teams, and our results have shown that not only skills, but also diversity, size, collocation, and full-time allocation matter when discussing agile team productivity.

Teams should be aware of the influence and magnitude of turnover, which has been shown, in most cases, negative for agile team productivity. Turnover has a disruptive effect on teams, becoming a particular challenge to self-organized teams. Project managers and team members should learn how to recognize the signs of disruptive conflicts to prevent productivity threats. Teams should also invest time exploring different modes of conflict management to keep issues under control. Human resources processes may be helpful in helping teams on solve turnover and subsequent team design issues.

Our results also suggest that motivation in the agile context is slightly different from the general view of motivation in software development. Some motivators become more important in agile teams, as well as new motivators emerge. We also noticed the importance of the context to establish a cause-effect relationship between agility and motivation. We call for research focused on a better description of context variables that capture relevant information to support motivation analysis in light of theoretical models. Finally, we found a relationship between past experience and motivation. Agile development seems to be more motivating for teams that worked before with other methods.

Finally, there are many possible directions for future research based on our results. Identifying productivity factors in a sociotechnical system is a challenge, but it should not be neglected. Research on productivity monitoring in agile teams may help the teams learn more about their own capacity to deliver and work as a team. Team members should be educated to understand and cope with productivity factors on a daily basis because they are self-managed. This is a shift from a self-organizing to self-managing capability that should be better explored. Likewise, researchers and companies should investigate appropriate strategies for inter-team coordination that consider agile team adaptivity and continuous delivery. Teams pace and priorities are important properties to be considered in this modeling. More research is needed to identify links between theoretical teamwork components to establish enlightening cause-effect relationships that help keep or improve agile team productivity.

6.4.6 Limitations

There are a number of limitations to this study. First, qualitative findings are highly context- and case-dependent [Patton, 1999]. Three kinds of sampling limitations typically arise in qualitative research designs: cases that are sampled for observation (because it is rarely possible to observe all situations); time periods during which observations took place (problems of temporal sampling); and selectivity in the people who were sampled either for observations or interviews or in document sampling. In pursuit of a trustworthy study, Lincoln and Guba [1985] proposed four main characteristics to which a qualitative study should pay attention: credibility, transferability, dependability, and confirmability.

To promote *credibility*, we adopted well established research methods and developed an early familiarity with the organizations culture through preliminary visits. Although we have used a purposive sampling of informants, we tried to include as many participants as possible from each team, considering similarities, dissimilarities, redundancies, and varieties to acquire greater knowledge of the wider group. We also triangulated data from three different qualitative sources: interviews, direct non-participative observation, and retrospective's documentation. Interview data were our primary indicators of productivity factors. The other two sources, nevertheless, influenced the emergence of the main factors in a significant way.

Credibility of a thematic synthesis also considers how well codes and themes cover data, i.e., no relevant data can be inadvertently or systematically excluded or irrelevant data included [Cruzes and Dybå, 2011]. We analyzed codes and grouped them systematically, adding more companies and different data sources to the analysis. We frequently referred to the data to ensure that codes were representative and check the relationship among codes and themes.

Confirmability is concerned with how the extracted data are coded and sorted and whether various researchers and experts would agree with the way those data were coded and sorted [Lincoln and Guba, 1985]. In this study, two researchers coded the data and agreed on each piece before adding them into the NVivo 9 tool for information classification.

Dependability concerns data stability, the degree to which data change over time, and adjustments made in the researchers' decisions during the synthesis process [Lincoln and Guba, 1985]. We described the changes that occurred in the companies, which helped us find some productivity factors. Cruzes and Dybå [2011] suggest complementary coding methods and establishing an "audit trail" that will allow an external reviewer to examine the processes whereby data were extracted and coded. However, due to the non-disclosure agreements, we did not provide the audit trail for external researchers, but only those participating in the research.

Transferability refers to the extent to which the findings can be transferred to other settings or groups [Cruzes and Dybå, 2011]. To promote transferability, we described the selection and characteristics of each case, including context and settings, data extraction, and synthesis process, as well as quotations with our major findings.

Another possible limitation is that we based much of our data on team perceptions. Once one productivity issue is solved, teams hardly notice its impact. For instance, all three projects substantially reuse software that is an acknowledged productivity factor [Basili, 1990, Mohagheghi

and Conradi, 2007]. However, teams had reached a good reuse level; its benefits were perceived, but with much less intensity than other factors emerging at the time of the study. To reduce the impact of this effect, we observed and interviewed teams over a period of six months, which allowed us to study the phenomena from different viewpoints as they emerged and changed.

Finally, the IPO model has served as a valuable guide for researchers [Mathieu *et al.*, 2008], where the input/output relationships are an important first step in any research program [van der Vegt *et al.*, 2010]. However, due to its limited and static perspective on team effectiveness and the dynamic processes that underline it [Kozlowski and Ilgen, 2006], IPO-style investigations may be more of the exception than the rule in modern-day organizations [Mathieu *et al.*, 2008]. Future studies should thus explore other contemporary theories that attempt to explain teamwork effectiveness, performance, and productivity in a more dynamic perspective.

6.5 Summary

This chapter presented our preliminary results that address productivity definition and factors in agile teams, i.e., RQ1 and RQ2. We claim that our main contribution are **C2. Rationale on productivity definition in the agile methods context**, **C3. Empirical verification of agile team productivity factors**, and **C4. A framework of agile team productivity factors and their impact, to be tested**, as mentioned in Section 1.6. As a minor contribution (C3.1.), we also added, to the literature, empirical knowledge on motivational factors that might impact agile team productivity.

Chapter 7

Productivity monitoring and measurement in agile teams

This chapter presents the last study of this thesis, which explores agile team productivity monitoring and measurement. We here describe the research method, results, and discussion of main findings related to the monitoring processes and usefulness of metrics.

7.1 Research method - Action Research

To answer Research Question 3, *How to monitor productivity factors, considering agility and adaptability? How do agile metrics support productivity monitoring?*, we chose the action research method. We aim at helping organizations to develop and sustain high performance agile teams. Our research goal is to develop a productivity monitoring approach for agile teams, helping them to reflect on their processes and products through action research.

We decided to adopt this research methodology for three reasons:

- The action research cycle fits into the performance measurement system development cycle (see Section 3.2.3). This similarity might help the researcher to deploy the productivity monitoring method and, in parallel, to study the phenomenon in the same cycle.
- I am currently working for Company D. We identified that the company needs help to handle productivity monitoring and measurement processes and also to evaluate their usefulness from a practical perspective.
- We wish to investigate monitoring processes that are not explicit in the studied organizations. Practitioners are not willing to discuss them with external people. We believe that researchers are only able to understand these elements if they work closely with practitioners.

In fact, we identified from Research Questions 1 and 2 that, despite companies consider productivity quite important, they do not adopt any formal approach to monitor it at the team level. Thus, we have a practical question from industry and also a need for more research studies in agile team productivity – which fits into the action research approach.

We thus aimed to conduct a participatory action research in industry to:

- identify specific productivity problems and goals in the company, particularly in a set of projects;
- assess the company's context, as well as teams and individuals in terms of teamwork and other factors that influence productivity in agile teams (we created the research protocol based on our conceptual framework presented in Chapter 3);

- propose (based on the literature) and collaboratively select a group of appropriate productivity metrics for the company context (based on process, product, personnel, project, and organizational goals);
- observe metrics usefulness throughout time and review them according to the team adaptation needs;
- report learning provided by each metric according to the teams' opinions and considering different dimensions;
- assess company, team and individuals to uncover hidden aspects of the team productivity monitoring.

We adopted **participatory action research** [Baskerville, 1999] because it allows researchers to get deeply involved with the organization as a group and engages the practitioners directly with research questions. Participatory action research was possible in this project because I worked for Company D during the last 10 months of this thesis research.

Figure 7.1 summarizes the design of Phase III, describing all phases and steps of our action research. We also explain the relationship between the research steps and the agile project releases. We tried to align the action research cycles with the company's project releases. However, we cannot assume that they will be always synchronized.

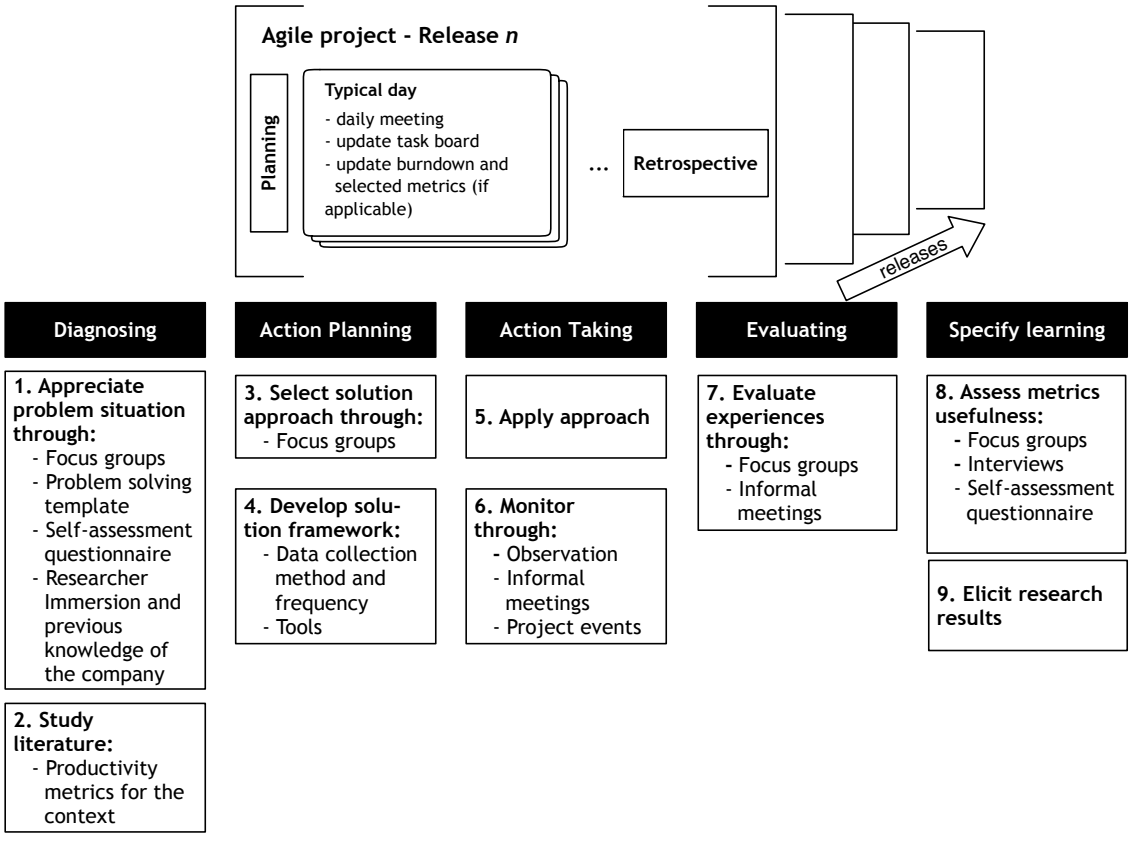

Figure 7.1: *Action research design*

1. Appreciate problem situation. We first conduct a diagnosis of the organizational, project, and team situation. Then we need to specify a specific problem to solve in a practical perspective, which is also interesting as a research question.

2. **Study literature.** Based on the company and team context, we study specific literature to synthesize possible directions of action. Whenever possible, our proposed conceptual framework (see Section 6.1) and literature review (see Chapter 3) should guide this activity.

3. **Select measurement/monitoring approach.** We engage the relevant literature, clarifying issues and identifying existing theoretical frameworks of relevance. Based on the literature and knowledge of the company, we define possible measurement and/or monitoring approaches to the company problem. In a collaborative process, the company and researcher select a solution approach to follow.

4. **Develop solution framework.** Researcher and company work together developing a solution, which might comprise the identification of actors, processes, measurement, and evaluation of the selected productivity monitoring and/or measurement.

5. **Apply approach.** Action is taken, i.e., the solution is deployed. The researcher remaining aware of his/her particular theoretical perspective throughout the process.

6. **Monitor.** The actions are monitored in terms of research interest [McKay and Marshall, 2001]. This step comprises problem solving and research.

7. **Evaluate experiences.** The researcher evaluates the intervention effect in terms of research questions. If the research questions can be answered or satisfactorily resolved, or in some way illuminated or even reframed, the researcher exits from the organizational setting. Otherwise, we amend our plans and designs to seek further explanations [McKay and Marshall, 2001].

8. **Assess metrics/monitoring usefulness.** Establishing usefulness of results in the problem situation supports the impartiality of the research and creates a baseline upon which the results might be transferred [Baskerville and Wood-Harper, 1996]. Experienced usefulness is the pragmatic basis for evaluating collaborative action research [Iversen *et al.*, 2004]. We assess the measurement/monitoring usefulness through individual interviews and group meetings (here considered project meetings, since this research is embedded in the real context, and the researcher is also a team member). Regarding the usefulness of metrics, the goal is threefold: i) describing perceived metrics' strengths and weaknesses in relation to productivity monitoring; ii) describing the perceived ease of use of the metrics, and iii) identifying for which development phase each metric is more appropriate.

9. **Elicit research results.** This last activity is recommended by McKay and Marshall [2001] to separate problem solving and the research process result. We can triangulate the team perceptions and their evolution as a team during the Action Research cycles.

7.1.1 Data collection and analysis

Action research is situational and bound by context [Susman and Evered, 1978]. Considering the importance of context to investigate how team members define or understand their situation, leading to possible actions, we improved our protocol to gather company context. In Phase II, we described the context of three companies for our case study. The challenge became stronger in Phase III as we were studying just one company/team for 10 months, trying to grasp their problems and need for change in a systematic way. The goal was not to contrast different companies, but to deeply understand a single team context and productivity and follow their improvement over time.

We thus explored the company context using the *levels of context* proposed by Kruchten [2011] (see Section D.1). It considerably improved our previous protocol to describe context applied in the multiple-case studies (Phase II).

Data were continuously collected, analyzed, reported, and revised throughout the project. Many of our protocols were created throughout the research cycles, because team productivity issues often change, demanding different approaches to gather information. Table 7.1 summarizes the data collection methods we chose to gather data during the three research cycles.

Table 7.1: *Data collection methods used on AR cycles*

Data collection method	Cycle 0	Cycle 1	Cycle 2
Interview	X		X
Informal conversation	X	X	X
Iteration retrospective	X	X	X
Project retrospective	X		
Field notes	X	X	
Project meetings	X	X	X
Observation	X	X	X
Emails	X	X	X
Project metric	X	X	X
Product metric	X	X	X
Teamwork metrics		X	X

We gathered data on *teamwork* using a well-established survey proposed by Campion *et al.* [1993] (Section D.4). We gathered *product metrics* aiming at understanding in which phase the product is, relating this information to business goals and, consequently, to measurement goals and adaptation needs. Utterback's and Klepper's models of product life cycle [Klepper, 1996, Utterback and Abernathy, 1975] supported this description.

We created an *interview* protocol with open-ended and close-ended questions (using a predefined scale) to evaluate team members learning using productivity monitoring and measurement processes. We first considered studies on the Technology Acceptance Model (TAM) [Venkatesh and Davis, 2000]. The TAM model is extensively adopted to assess technology usefulness from two perspectives: i) perceived usefulness, and ii) perceived ease of use. However, this model is overly broad and guided by technology acceptance. Our goal requires a more specific tool to support the evaluation of productivity *monitoring* and *measurement*.

We then found studies on metrics acceptance [Umarji and Emurian, 2005] and measurement program success [Staron and Meding, 2011b]. Umarji and Emurian [2005] describe a detailed protocol to evaluate software metrics usefulness based on the TAM Model. However, it describes metrics in an organizational context very different from that we were studying. The proposed assessment is based on the idea that companies create a measurement program for teams adhere. However, agile teams usually create their own measurement system, based on their context and problems. So, we adapted all questions related to this difference between traditional and agile teams. Staron and Meding [2011b] describe success factors for software measurement programs. Some success factors overlap Umarji's protocol, then we did not repeat them. We just added questions related to i) the incremental implementation of metrics and ii) the congruence of measurement goals with business goals.

We also added questions derived from our propositions for agile team productivity monitoring and measurement (see Section 3.4.4) to the interview protocol. Our purpose was to determine whether team members learned from metrics or not. For instance, *Proposition 1* claims that *productivity measurement should be used predominantly for learning, short-term management, and long-term improvement. As it is a dynamic measure by nature, it may differ by products and the technology used. The set of metrics may vary at different stages of a product life cycle and should be developed in a process-oriented bottom-up approach.* We thus added questions to evaluate how metrics are introduced in the project, if they change, if they relate to business goals etc. Appendix D details the instruments we are using to collect and analyze data in this research phase. We interviewed five team members to determine monitoring and measurement usefulness based in this protocol.

We decided to use field notes and observation with an opened protocol. We took notes from what we observed on meetings, regular sessions of work, emails, and informal conversations. Photos from retrospectives[1], and also project, product, and teamwork metrics were added to enable data collection triangulation. As mentioned, metrics were also evaluated from a learning and adaptation perspectives. So they were not only input for triangulation, but also subject of study.

Finally, most data analysis were based on steps of thematic analysis, the technique we adopted for our multiple-case studies, particularly *text reduction into codes* and *exploration* (see Section 6.2.2). Quantitative analysis was required to interpret some metrics throughout the study, but it just involved very simple descriptive statistics.

7.1.2 Quality procedures

Davison *et al.* [2004] proposed a criteria to guide and evaluate action research projects to avoid pitfalls described by Baskerville and Wood-Harper [1996]: (1) lack of impartiality of the researcher; (2) lack of discipline; (3) mistaken for consulting; and (4) context-dependency leading to difficulty of generalizing findings. We use the criteria to evaluate our research:

1. *Roles* - What are the researcher and practitioner roles and how do they develop over time?
2. *Documentation* - What data are collected to support the problem solving and research goals; how are these data collected; and how is data quality ensured?
3. *Control* - How is the researcher-client relationship established; who exercises authority over the process; and to what degree are formalized control mechanisms adopted?
4. *Usefulness* - How is usefulness of the solution established in the problem situation?
5. *Theory* - How are frameworks used to support the study; and how are the results subsequently related to these frameworks?
6. *Transfer* - Under what conditions can the results be transferred to or adapted in other contexts?

7.2 Empirical results

We describe, below, results from the action research analysis, presenting some quotations and other evidence that support the findings.

7.2.1 Cycle 0 - June/2012 to August/2012

Diagnosing

Company D is a multinational company working with cutting-edge software technologies and processes, with a local presence in Brazil. The organizational structure and coordination are primarily horizontal [Galbraith, 1973], and it is flatter than all other companies in our study. The company is one of the most influential players in the market of agile methods, and disseminates its culture based on sustainable business, technical excellence, and social justice. However, as a consultancy, the projects are not only immersed in its own culture, but also in the customers' culture, which may differ from project to project. Table 7.2 summarizes information for Company D.

Project D is a web-based solution for digital content, built upon a content management system. The client is a national company that is investing in becoming an Internet company, crossing the gap between print and digital media business. We started the project with the 18-people team described on Table 7.3.

[1]Some details on photos are blurred to assure company anonymity.

Table 7.2: *Company D Profile*

Characteristics	Company D
Company business	Multinational company working with cutting-edge technologies and processes, focused on delivery, consultancy, and products.
Company structure	Horizontal
Number of IT employees	2000 worldwide and 200 in Brazil
Maturity	In Brazil, for 2,5 years. Over the world, > 15 years
Level of innovation	Considered innovative because of cutting-edge technologies, early adopter mindset, and risk-taking projects.
Culture	Values and beliefs based on sustainable business, technical excellence, and social justice.

The team fully adopt agile practices such as *Continuous integration, Daily meeting, Energized work, Pair programming, Real customer involvement, Shared Code, TDD, Planning game, Retrospectives, Slack, Stories, Team continuity*, and *Whole team*. Another practices were partially adopted due to client or project constraints, such as *Sit together* (distributed project), *Daily deployment* (the client infrastructure does not allow it for every case), *Code & Tests* (there was some documentation to deal with geographic distribution), *Negotiated scope contract* (redevelopment of an existing product hampers scope negotiation).

The project is an existing system redevelopment, aiming at promoting product innovation. From a product life cycle perspective, the product current version was mature. However, due to market competition, aspects as usability, performance, reuse, and new market segmentation turned into important product innovation drivers. Figure 7.2 illustrates the product phase from a life cycle perspective - from maturity to decline - investing on the product redevelopment to achieve needed innovation.

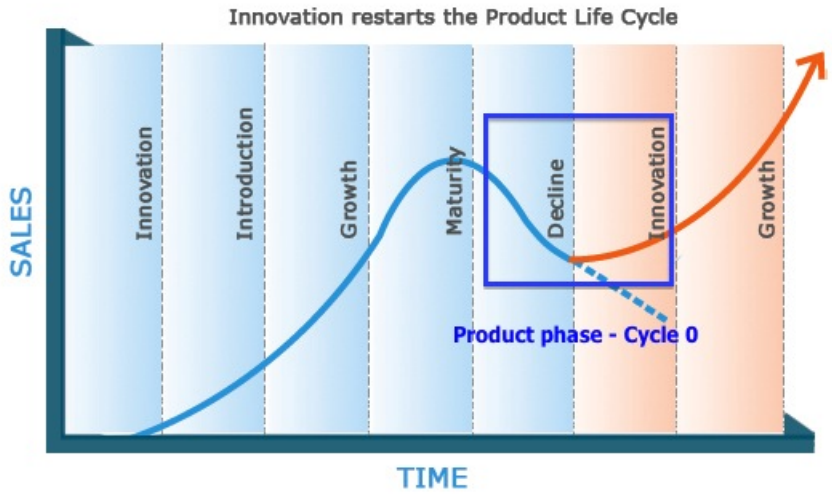

Figure 7.2: *Innovation restarts the Product life cycle*

One of the goals was to develop the product on top of a new corporate system that provides content management facilities. Because the new product aimed at substituting the older one, its

Table 7.3: *Project 4 Profile*

Characteristics	**Project 4**
Project description	Digital media and content
Project start	December/2011
Team composition	8 full-time developers, 3 business analysts, 3 quality assurance analysts, 1 iteration manager, 1 project manager, 2 product owners (Total = 18 members)
Team distribution	Team members are geographically dispersed in the same country, in two different states. Some team members are foreigners.
Staff turnover	50% - Considered high by the project manager
Technologies	Ruby on Rails, Sinatra, MongoDB, RESTFul APIs, Vanilla Javascript, jQuery, Rspec, Jasmine, Cucumber, Monitoring with Jenkins, JMeter, Selenium, Capybara and NewRelic.
Architecture	One single system integrating with many other APIs from the customer side.
Main non-functional requirements	Availability, Performance, User Experience
Reuse	High - Components and other systems, such as a Content Management System
Requirements stability	Low stability
Product criticality	Product failure does not hurt people. It causes loss of money.

deployment was not iterative. The client had decided to turn off the old system at a time, and launch the new system in the following day. The team was then working on software development with no single delivery for months.

Figure 7.3 is a photo from the 18th retrospective, bringing evidence of the staff turnover effect on team motivation and productivity. They felt tired sometimes, because they felt incapable of making an effective change.

Figure 7.4 illustrates problems the team was facing in the 20th iteration, such as demotivation, perceived through complains, and excessive focus on the team velocity metric, instead of measuring value.

Team members defined team productivity in a similar way:

> "Team productivity is rather a subjective definition [...] for me, it is to deliver value in a consistent and frequent basis" *(Senior business analyst 1)*
>
> "It is to respond to changes as fast as we can. It's reducing the gap of time between what customer required and what we delivered" *(Developer 1)*
>
> "It is how fast we can deliver value, from the moment the customer has an idea, transform it into something related software, until the delivery" *(Developer 2)*
>
> "Deliver high quality features as soon as possible into production" *(Quality assurance analyst 1)*

After diagnosing the problem situation, we start next research step by studying the literature. Some of the issues were well explored in our previous studies (see Chapters 3 and 6), as staff turnover, inter-team coordination and motivation. Other factors such as trust, communication, and cultural adaptation strongly emerged due to the distributed project nature, which also depends on two different companies to succeed.

Figure 7.3: *Retrospective of Iteration 18*

As Table 7.5 shows, there were many factors to tackle. Management decides to prioritize some of them, particularly all team factors, and some process, project and product factors. However, due to the project constraints, most of actions were taken in the second cycle, after the first official delivery.

From the problem situation, we identified three important goals for the team: *efficiency*, *speed* and *timeliness*. Efficiency and velocity because the scope was bigger than originally planned and the client needed a product delivery in the original time planned. The team needed to figure out actions to achieve this timeliness by improving their efficiency and velocity.

Efficiency is one dimension of team productivity. Increasing efficiency without maintaining effectiveness is a problem, because while *efficiency* represents how well the "resources" of software development process are utilized, *effectiveness* represents the degree to which desired results are achieved (see Chapter 3). The goal must be optimizing both to achieve productivity. In Section 3.4.2, we presented agile team productivity metrics. Efficiency can be related to waste reduction, and speed to velocity or aceleration.

In lean thinking, waste detection and removal is a way to improve efficiency in a system [Poppendieck and Poppendieck, 2003]. Waste is everything that does not add to the customer value of the products [Ikonen *et al.*, 2010]. In general, seven types of waste were identified [Poppendieck and Poppendieck, 2003]: partially done work, extra processes, extra features, task switching, waiting, motion, and defects.

In literature we found two studies focused on how to monitor waste from a lean thinking approach. Ikonen *et al.* [2010] used semi-structured interviews to identify sources of waste in a project.

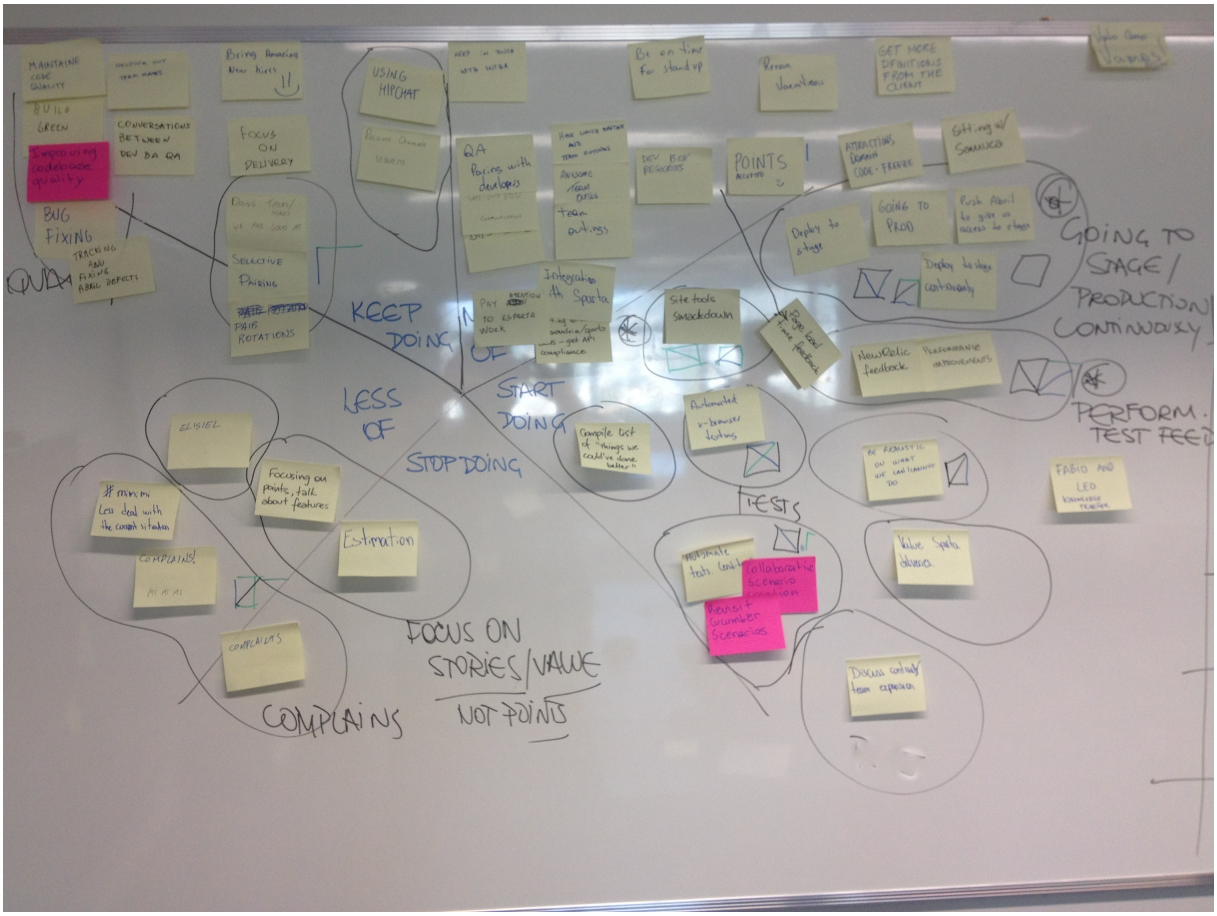

Figure 7.4: *Retrospective of Iteration 20*

Petersen [2012] presents a palette of indicators to assess waste in the maintenance process based on four lean indicators (inflow, flow, lead time, and workload).

Finally, velocity is a metric for speed used for predictability and delivery capacity estimation [Beck and Andres, 2004]. Acceleration complements velocity by indicating relative changes in velocity - when velocity increases, acceleration is positive (as mentioned in Section 3.4.2).

Action Planning

The team was not using lean metrics in the project, and they did not have time to implement big changes due to the tight schedule. The team decided to have a focused retrospective, which is a session to investigate roots from a particular problem and decide actions. They discussed possible 'out-of-the-box' ideas to improve efficiency. They also needed to evaluate how to respond to the zero-bugs requirement that client was asking. After a 1-hour meeting, the team decided to commit to the client goal based on efficiency, with these actions:

- Avoid unimportant meetings;
- Proactively monitor waste in the process;
- Being selective on pair programming: forming pairs considering best fit to solve a problem and use solo programming when the task is easy;
- Monitor the number of bugs opened to keep them low, even when bugs were not important;
- Work more than 8 hours/day for some weeks.

The approach was divided in three parts: *team members' self-monitoring*, *project speed*, and *process efficiency* measurement. *Self-monitoring* was based on the team commitment to not attend project unrelated meetings, working more than 8 hours/day, reducing pairing, and proactively monitoring sources of waste to improve *process efficiency*. There was no specific tool to identify waste, just an understanding of the seven types of waste proposed by Poppendieck and Poppendieck [2003]. *Speed* was measured in two ways: by the *velocity* metric (see Chapter 3), visualized in a burnup chart, and by a list of opened bugs. The purpose of bug monitoring was to visually show how much bugs we had to fix, apart from their priority or impact. Figure 7.5 shows the visualization of both metrics.

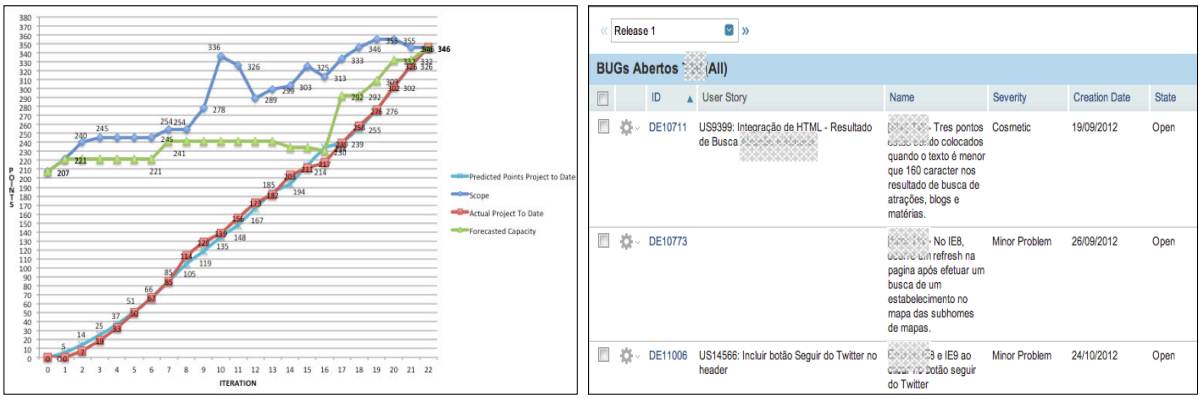

Figure 7.5: *Metrics supporting speed monitoring (a) velocity (b) list of opened bugs*

Both metrics were built upon the client tracking tool that includes stories, bugs, and many other data from an agile project. The team was using the burnup chart from the project beginning and the bug metric was created in the context of this action.

Action Taking

After committing to the action plan, the team adjusted their mission to focus on efficiency, speed, and timeliness immediately. The team was committed to these goals before, but at this moment the expectation was much stronger. The client turned off some automated test suites to speed up development, provoking team demotivation. On the one hand, the client decided to buy a huge technical debt to enable timeliness. On the other hand, the team disagreed with the decision as they would work in the codebase after the first release.

At the end of each iteration, the team kept presenting the achieved results, but stressing more the velocity metric (number of points delivered). The client was inspecting results and project velocity metrics.

Evaluating experiences and specifying learning

Productivity monitoring was not very active during this cycle. The project managers changed in this period and had not yet devoted full attention to monitoring and measuring team productivity. We could evaluate experiences through in-depth interviews, project meetings, and iteration retrospectives.

Removing meetings showed results, but for some meetings (organizational meetings, knowledge sharing meetings etc.) it might have represented a slack reduction, that could cause stress if maintained for a long time.

Team members expressed both positive and negative aspects of metrics in their context:

> "Velocity was the most used metric in the project, it was the most visible metric. Velocity is a way to show that some scope, measured in points, is passed from one iteration to the

next [...] it also shows that there is always a variation during the project, which is natural in projects [...] Sometimes it is unrealistic. For example, a runner cannot maintain his maximum speed in the entire marathon. But this is demanded from the development team [...] Slacks would help us to overcome this [problem]. Some waste-cutting actions also cut the slack to improve velocity." *(Senior Business analyst 1)*

"Velocity does not make much sense really. We can't estimate, we can't know the future [...] a 3-point story becomes an 8-point story, because there are so many things behind. And then the client focuses only in points and starts to ask 'why does this story is not 3 points' [...] they used it against us. There should be a reason for velocity, but in this context, in our project, with this client, it didn't work. We didn't have a chance to adjust our velocity, we tried to deliver according to this number. We are focusing too much in these points, instead of value or quality [...] It didn't reflected reality in our team." *(Developer 1)*

Table 7.4: *Experiences with agile team productivity monitoring - Cycle 0*

Dimension	**Goal**	**How to monitor**	**Evaluation**
Process	Efficiency	Proactively reducing waste	(-) Some types of waste reduction may represent a necessary slack for teams
	Speed	Velocity	(+) Show natural project variations (e.g.: S curve) (-) With the intrinsic estimation uncertainty, the metric easily lies. (-) If the maximum team speed is taken as their normal speed, team will burn out.
		List of bugs opened	(-) Reinforce the lack of bug prioritization. There is no zero bug system, bugs need to be prioritized.
Product	Timeliness	Iteration planning meeting: regular meeting to compare planned and obtained results, and plan the next iteration.	(+) Informal monitoring that makes timeliness visible.

The overall learning was that productivity expectations were clearly misaligned. While the team was considering productivity as delivering value frequently, the client was demanding efficiency. The team decided to commit to the client's productivity expectation, delivering in time, with the desired efficiency, while neglecting other aspects such as quality. The negative outcomes from this misalignment emerged: high staff turnover (50% of team members left the company), very low team morale, team not willing to work with the client again, and an evident cultural misfit between the ways which both companies work. Such misalignment seems to cause dysfunctional measurement behaviors [Austin, 1996], since measurement indicators were high, but true team performance was low.

We thus confirmed that studying agile team productivity in this project would be helpful for the team. Therefore, the diagnose-cause-intervention chain and assessment of the agile team productivity would be central to both objectives of our action research: practical problem solving and theory development.

7.2.2 Cycle 1 - September/2012 to December/2012

Diagnosing

The first project release happened and a next project phase was beginning. We decided to map the most painful team and project issues through a project retrospective to be able to keep the engagement without losing more team members. Before this decision, there was a discussion by email between leadership members on running "group therapy sessions". One member suggested that listening to too many complaint would make people feel even worse, negatively influencing those members that did not have specific complaint. We suggested that conflict resolution and complaints management is important for teamwork and team productivity, and that receiving feedback from our client is a gift, even being negative [Barlow and Möller, 1996]. We finally decided to have the project retrospective, facilitated by a company expert on people management.

Through different sources of evidence, such as the project retrospective, past project meetings, informal conversations, and existing metrics, it was possible to delineate the team and project most painful concerns. Table 7.5 describes the main issues identified in the project retrospective (results partially transcribed from a company document).

From a product perspective, business was demanding actions to sustain audience, one of the key metrics they use to evaluate product success. These actions are related to the product innovation. Timeliness was thus mandatory, because business areas needed the product features to leverage their marketing strategies and campaigns.

From a project perspective, new perspectives emerged: the team could deliver more frequently and the client was more opened to understand prioritization based on value. These changes were progressively more noticeable by both sides (the team and the client) as the project was progressing.

Another important change is that part of the team moved to the client side, aiming at supporting communication, trust, and team building. In this process, the client decided to include six more employees in the development team playing technical roles. Team size increased and, from a team development viewpoint, they faced a storming phase with conflicts, reconciliation, and team evolving by setting new norms [2].

Action Planning

Most of the identified problems were related to human issues. The team needed to address communications problems, trust, motivation, a shared understanding with the client, and a cultural adaptation. We found, in our previous multiple-case studies, that team management factors were the most frequent in the three evaluated companies. We could then interfere with the context, since this is one of the action research goals and also because I was a team member, which facilitates and legitimizes action proposals.

At first, we decided to investigate patterns of trust, because we already identified motivation and staff turnover patterns in the team. One of the most interesting studies on trust was described by Moe *et al.* [2008]. They summarized key smells that help teams to identify *trust* problems in co-located and distributed teams.

We also performed our own trust investigation with multiple company D employees. It was a short qualitative case study in which we adopted the same protocol used by Dorairaj *et al.* [2012]. We interviewed 10 people[3] and discussed preliminary results in a workshop. We gathered some opinions and confirmation of previous research on trust problems and possible solutions [Dorairaj

[2]Team development phases are described in our conceptual framework presented in Section 6.1

[3]Specific study results are out of the scope of this thesis and are under preparation for future submission.

Table 7.5: *Company D productivity factors and issues identified from data sources*

Negative Factors	Themes/Issues identified	Rational from multiple sources of evidence
Personnel	**Low motivation**	Team is clearly demotivated. Lack of client acknowledgment is a strong factor.
		Need to filter bad client inputs at the management level, not letting team moral goes down
		Need to be more pro-active
		Retrospectives turned into whining sessions
	Lack of trust	Meet again people on the client side, because they have changed. Lack of trust hampers any alignment endeavor
		Need to create a common mindset instead of us against them
		Low C-level involvement and lack of communication channel
		Lack of involvement/communication with other IT teams and Product team
		Need to set the right expectations for team and stakeholders
		Lack of influence on client decisions
	Staff turnover	Try to not loose more team members
		More smooth rotations in the project - staff management.
Process	**Scope creep**	Scope out of control, forcing changes on process (e.g.: being selective on pair programming to increase efficiency)
	Lack of adequate infrastructure	Bad infrastructure to enable continuous delivery
		Bad infrastructure to enable distributed work
	Poor inter-team coordination	Poor inter-team communication and coordination
		Spent too much time trying to remove blockers from other dependencies, that time we could use to understand better what out client needs.
	Trouble with learning	Team unable to learn from retrospectives, because action is not happening
		Stop complaining, focus on solutions
	Low quality effectiveness	Dysfunctional measurement of quality create extra and unnecessary pressure instead of focus on delivering value
		Lack of automated testing
Project	**Scope creep**	More scope, same deadline. Low quality, creating tech debt.
	Lack of ownership	Need to have more ownership about project decisions
Product	**Excessive focus on efficiency**	Need to focus more on value being added to the project, instead of focusing on velocity
	Dysfunctional quality measurement	Focus on wrong metrics, such as number of bugs opened, instead of focusing on measuring defect removal efficiency of prioritized bugs
Organizational	**Cultural misfit**	Both sides think they are very different from each other.

et al., 2012, Moe *et al.*, 2008]. For instance, Moe *et al.* [2008] proposed actions from empirical studies that may help teams to succeed on the problem solving. They recommend actions such as i) investing time in one or several face-to face meetings with the team members; ii) communicate expectations early and establish initial rules; and iii) create a common understanding of the work process and how to cooperate in this process. Our internal case study pointed out similar results.

We found in our previous study [Melo *et al.*, 2012] that agile individual and team *motivators* are slightly different from traditional motivators on software development, based on a systematic literature review and multiple-case studies. Such results may help us to better investigate underlying causes of problems of motivation and solve those that are hampering the team productivity.

From a project perspective, another important decision was to not highlight the velocity metric in the first weeks of project. Team members were stressed by the results of the velocity measurement that was used to control and drive the project release 1.

Finally, the team needed to map all technical debt created to deliver release 1 and prioritize it in the requirements backlog. Cunningham [1992] coined the technical debt metaphor in his experience report presented on OOPSLA'92 to describe "a situation in which long-term code quality is traded for short-term gain, creating future pressure to remediate the expedient" [Brown *et al.*, 2010]. The plan was to regularly run sessions to elicit, discuss, and prioritize technical debt from a team perspective, and then negotiate it with the client. Paying technical debts is essential to keep long-term productivity [Tom *et al.*, 2013].

Action Taking

To create a common understanding of the work process and of how to cooperate in this process, we had three workshop sessions to create and review the process; one in each project location and another one when new team members joined (team storming phase). We created a virtual wall to synchronize the physical walls of both locations. The three walls represent the same Kanban system defined in the workshop sessions. The team defined explicit criteria to define when a certain story should move to the next lane in the Kanban. Figure 7.6 shows one of the physical Kanbans.

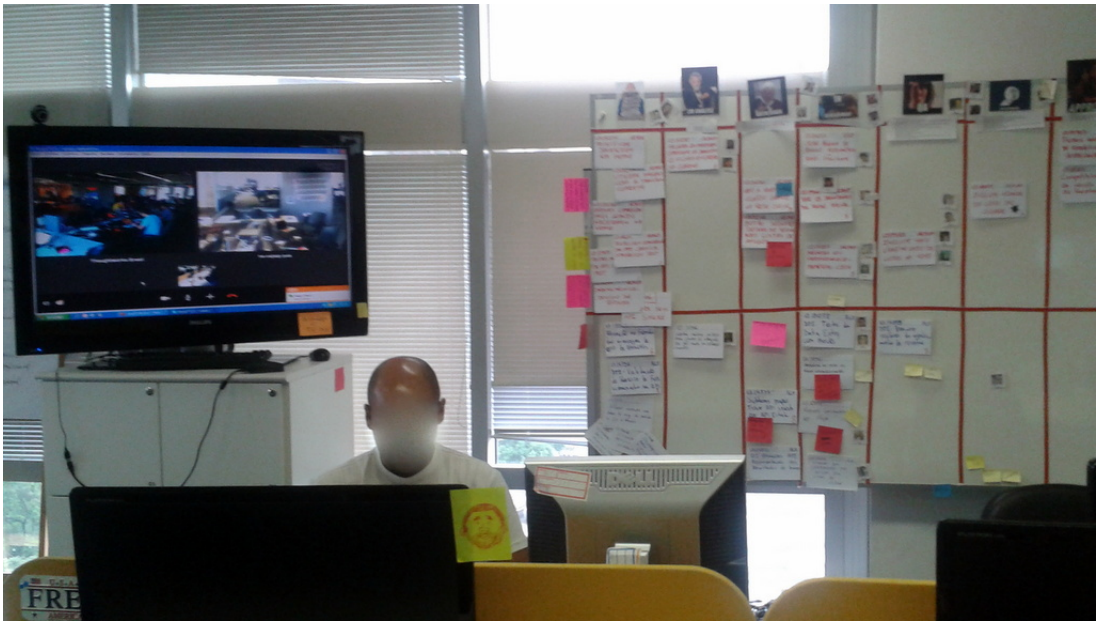

Figure 7.6: *Physical Kanban*

To enable trust, there were many small actions. Management spent considerable time on 1:1 meetings with the team members, identifying personal viewpoints, problems, and defining short-term goals. Management also followed up those actions to make sure they were effective. Management also spent time with the client leaders to understand their culture and processes. The project

manager decided to ask a vote of confidence from the client to improve the relationship[4]. The team was able to get funding to travel more between project sides, so the team members could finally spend some time together and develop a relationship. Telepresence was also an important action to make both sites visually connected, as shown in Figure 7.6.

To motivate team members, the first natural action was to deliver software frequently. The system was entirely redeveloped and it was in production, so there was no reason to not release software every one or two weeks. The team kept scheduling extra work activities, such as dinners, to foster and maintain integration. New team members joined the team (effect of turnover) and management set clear expectations of helping to improve mood and morale.

The team discussed technical debt and created a graph, evaluating each one by pain and effort (Figure 7.7). Sessions were informal and results created on a shared-decision process in each all team members participated in the prioritization. A team member explained:

> "We monitored tech debt on the wall. We mapped tech debts in a bi-dimensional graph evaluating value versus effort [...]. The more painful and easy to deliver the tech debt was, the more we tackled it [...]. We made decisions using a cutting line that separates those we want to solve in the next iterations" *(Senior Business analyst 1)*

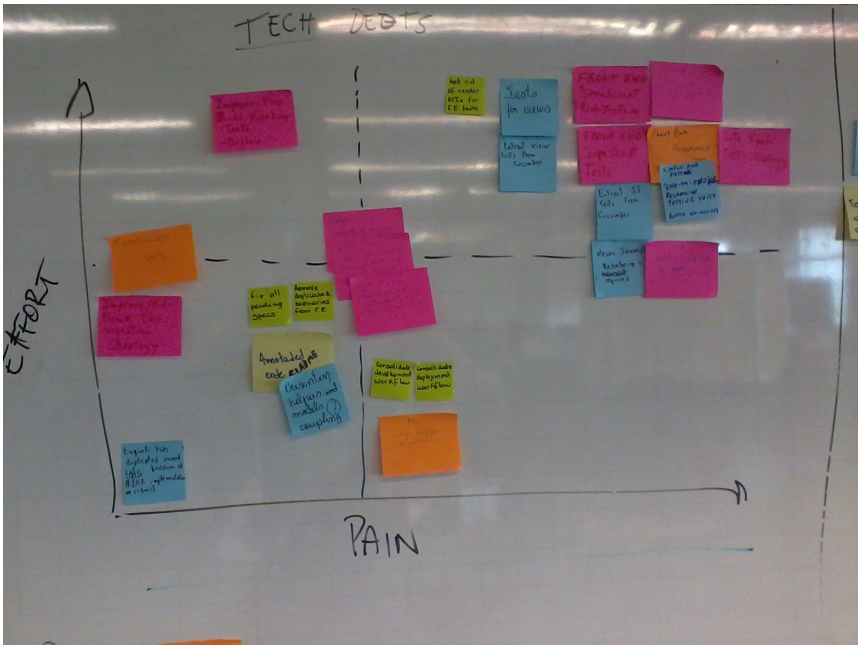

Figure 7.7: *Monitoring Tech Debt*

Evaluating experiences and specifying learning

In this cycle, it was clear that team productivity monitoring cannot be based only in quantitative metrics. Soft factors are hard, or even impossible, to measure. The team needs to develop other mechanisms to scan the environment and identify productivity issues.

We then resorted to qualitative indicators, which are useful when you cannot count, but just assess reality through opinions and perceptions. Good examples in this phase were trust and motivation factors. Such negative factors, or anti-patterns, can be monitored through team regular meetings, as retrospectives. These factors were not always the most voted for actions when the

[4]There were many discussions on trust building with thought leaders as Michael Sahota and Diana Larsen. Sahota told me: "Be the change you want to see", quoting the famous phrase of Gandhi. The rationale is that: to build trust, you need to first decide to trust someone and then expect some result.

team members made shared decisions on retrospectives. However, their influence is remarkable. One of the limitations was that just management was focused on the monitoring. We believe that a possible improvement would be sharing this responsibility with the entire team.

Technical debt was also qualitatively monitored. The practice helped to create a shared vision on the issues, not only inside the team, but also with the client. The technique was used for a short period of time and did not embody the team frequent practices. There was also a feeling that an automated solution could help to identify some types of tech debts faster, for instance, internal code quality degradation.

Timeliness was still monitored through iteration plan meetings, but it was gradually changing into a more frequent and organic monitoring in the daily standups, which involved the two project locations via telepresence. The client had not only the physical wall in his location, but also the digital wall that enabled fine-grained progress tracking.

Table 7.6 summarizes which anti-patterns we monitored, the actions the team took to solve them and an evaluation. We focused on the most important outcomes of this phase: trust, motivation, and technical debt monitoring.

Finally, one of the most important outcomes from these actions was that the team members stopped to leave the company, as happened in Cycle 0. This was a clear sign that actions were producing positive results.

7.2.3 Cycle 2 - January/2013 to March/2013

Diagnosing

From a product perspective, the client was satisfied with the product performance and its results in the market. The product was evolving in its life-cycle to an innovation period, in which creative ideas were welcome. The client was demanding quick responses to changes, with some predictability. Innovation, agility, and predictability were major drivers at this point. On the other hand, the team wanted to keep technical debt under control, to not allow short-term productivity because of the agility need.

From a process perspective, iterations became useless for a team able to deliver frequently. With an improved relationship and more interaction with the client, project status was clearer and activities were more focused on irradiating backlog reprioritizations. There was a common desire to make changes in the process to reflect more Lean principles [Poppendieck and Poppendieck, 2003, 2006]. However, as most of team members never worked with a genuine lean process, the team needed to invest on training. Learning, thus, was an important productivity driver. Discussion on process details was already happening because some team members did not agree with all quality verification steps.

As the team size increased it became clear to the management that there were miscommunication patterns going on inside the project. Project meetings as standups and retrospectives were too long because there were many details and context to share between sites. Cultural adaptation was still required to enable a consistent team development. After trust building, the next step would be fostering teamwork.

Campion *et al.* [1993] proposed a tool to evaluate teamwork in self-managed teams by measuring its main features. They developed a survey with 54 questions divided into 19 constructs, each construct representing a desirable feature of a team. They suggested that the set of tasks assigned to a team should provide a whole and meaningful piece of work. This allows the team members to see how their work contributes to a whole product or process, which might not be possible with individuals working alone. This can give workers a better idea of the significance of their work and create greater identification with the finished product or service. If the team workers rotate among a variety of subtasks and cross-train on different operations, they should also perceive greater variety in the work. Interdependent tasks, goals, feedback, and rewards should be provided to create feelings of team interdependence among members and focus on the team as the unit of performance, rather than on the individual. It is suggested that team members should be heterogeneous in terms of areas

Table 7.6: *Experiences with agile team productivity factors monitoring - Cycle 1*

Dimension	Goal	How to monitor	Actions	Evaluation
Personnel	Trust	**Monitor anti-patterns in a team [Moe *et al.*, 2008]:** • Lack of task-relevant background information to remote team members. • Decreased information exchange and feedback; • Competition and not cooperation; • Self-protection; • Doubt negative feedback from manager; • Relationship conflict; • Individual goals over group goals; • Team not self-correcting; • Not shifting workload among members; • Productivity and quality decrease.	• Identifying anti-patterns on iteration retrospectives and 1:1 meetings • Establishing short-term goals in 1:1 meetings and follow-up • Investing on team events out of the office • Funding traveling between sites	• (+) Triangulating retrospectives results and 1:1 meetings were effective to identify trust issues. • (+) Deriving short-term actions from 1:1 and following up showed significant relationship improvement between team members, thus trust. • (+) Events help to 'break the ice' and develop intimacy. Particularly useful when team members turnover happens. • (+) Traveling is essential to create team spirit, shared vision of practices, and to solve individual conflicts. Physical presence is not dispensable.
	Motivation	**Monitor anti-patterns in a team [Melo *et al.*, 2012]:** • Amount of changing; • Work/life balance; • Lack of challenging work; • Lack of teamwork and task identity; • Lack of feeling of progress/accomplishment; • Lack of opportunity to widen skill; • Poor management; • Lack of feedback (from colleagues, on goal accomplishment); • Lack of employee participation/involvement/working with others etc.	• Spending time to create conditions or remove blockers for frequent delivery. • Enabling sense of purpose and urgency when planning next deliveries (goal setting). • Fostering a culture of continuous feedback, remember to acknowledge good results.	• (+) It is crucial to enable frequent deliveries if a team that considers productivity as the ability of deliver continuously. Motivation was higher and some anti-patterns disappeared in feel weeks. • (+) Team members highlighted how motivated they were on delivering a feature planned with the team and the client, with detailed explanations on business reasons and goals for that feature. Team delivered faster. • (+) Specific feedback sessions deeply changed members' individual performance. It was confirmed through the company's annual performance review and observation.
Product	Quality	Technical Debt	Eliciting, discussing, and prioritizing technical debt through regular sessions, framing results in a graphic	• (+) Helped to create a shared vision of current technical debt. • (-) Seems to not work for a daily basis identification. Team miss an automated solution.

of expertise and background so their varied knowledge, skills, and abilities complement one another. Teams also need adequate training, managerial support, and organizational resources to carry out their tasks. Managers should encourage positive group processes including open communication and cooperation within and between work groups, supportiveness and sharing of the workload among team members, and development of positive team spirit and confidence in the team ability to perform effectively [Morgeson *et al.*, 2006].

Action Planning

The project was divided into two subprojects to improve communication. There were trade-offs involved: each subproject team would loose the others' context and maybe one subproject would be more interesting and exciting than the other.

The team decided to invest on Lean adoption by organizing workshops on both locations to create a share understanding of the intended process. They also spent time defining how to limit the work in progress (WIP) and improve flow. This change was not planned as a formal process, but it was rather an informal and organic change. The team members decided to run a small experiment based on root cause analysis of bugs to determine why some defects appeared and to confirm the need for quality verification steps.

To achieve product demand, it was important to create metrics to support predictability calculation. Metrics as throughput, lead time and WIP were necessary to the project (presented in Section 3.4.2). Management was responsible to create metrics using the same tool that provides the virtual Kanban.

Finally, we planned a self-managed teamwork assessment based on Campion *et al.* [1993] to investigate hidden aspects of teamwork productivity that did not appear in retrospectives, informal conversations, or 1:1 meetings.

Action Taking

The team was divided in two, but some members remained in both teams, as experienced business and quality assurance analysts. Management provided more space for the teams in each location, in which team members from a same sub-project sit together.

We created the cumulative flow diagram (CFD) to support management with lean metrics, a visualization technique discussed in Section 3.4.2. Figure 7.8 shows one of the subprojects CFD. The diagram supports the visual analysis of work in progress (WIP), lead time, and bottlenecks in the software development process. In parallel, the team organized the workshops on lean thinking involving practical exercises to stimulate the new process learning.

The teamwork assessment was divided in three steps: i) data collection, ii) data analysis, and iii) a session to discuss results and propose actions. Fourteen team members answered the questionnaire in the same week, spending 20–30 minutes each one. We assured anonymity by using a paper-based version and similar pens. We performed data analysis as described by Campion *et al.* [1993]. Scale for all questions ranged from 5 = "strongly agree" to 1 = "strongly disagree". We aggregated the average of responses across the fourteen members in each group. Figure 7.9 illustrates the assessment result in Kiviat chart.

Finally, some developers introduced a new static analysis tool called Code Climate[5]. Some team members started a discussion around quality metrics and they came up with a final decision. Developers and management worked together to buy the tool and scheduled a small session to explain the metric to all team members. The tool automatically scans the codebase looking for code smells, i.e., patterns of low internal quality. Every time the internal quality decreases, the tool alerts team members by the team chat.

[5] Available at https://codeclimate.com/

Figure 7.8: *Cumulative Flow Chart from one subproject*

Evaluating experiences and specifying learning

Table 7.7 summarizes main experiences and reflections from the team members after the action taking.

Experiences with the cumulative flow diagram (CFD) showed that its main strength is transmitting a sense of completeness when there is a delivery target. The team members considered the CFD an informational metric [Austin, 1996] that motivates members. On the other hand, some members described the CFD as a reactive metric that shows problems that already happened, for instance, process bottlenecks. In most cases, the team was able to identify bottlenecks before the CFD. Nevertheless, the CFD supported focused retrospectives by visually expressing WIP, lead time or bottleneck problems. The team members thus have additional data to investigate problems, instead of just perceptions or opinions.

Our pioneer initiative of measuring teamwork in the company generated multiple views. Some developers said:

> "Even though we were not clear on the meaning of each dimension, it generated some interesting discussions." *(Developer 3)*
>
> "We should do that more regularly" *(Developer 4)*
>
> "It generates some overhead because we first spend 20 to 30 minutes to respond it, and then we need a session to discuss results. [...] But I don't thinks this is a intrinsic problem of the measure, but of the way we did it" *(Developer 2)*

In fact, the protocol provided by Campion *et al.* [1993] did not elaborate much on how to communicate results for a team, or how to enable actions from it. One of the learning outcomes was that we need to invest time preparing the next assessment, sharing main concepts behind each survey question.

Discussion focus on lower results as *Workload sharing*, *Training*, and *Feedback/Rewards*. The team members felt that there were cases on unbalanced workload sharing due to the knowledge silos in the team. This might be explained by the not so high team heterogeneity. However, the team members highlighted the lack of pair programming involving different team roles, as developers and quality assurance analysts. They discussed also strategies to overcome silos when new team members join the team, such as an onboarding (meeting to explain overall system), to complement pair programming. These issues might be hampering the team productivity, but did not appear in previous discussions. Training and Feedback/Rewards generated diffuse discussions with no general

Table 7.7: *Experiences with agile team productivity monitoring - Cycle 2*

Dimension	Goal	How to monitor	Evaluation
Process	Leanness /Flow	Cumulative Flow Chart	(+) Sense of completeness when there is a delivery target (e.g.: CFD with expected scope). Informational metric that motivates. (-) Reactive metric that shows problems that already happened (e.g.: process bottlenecks). (+) Support focused retrospectives by visually expressing process problems.
Product	Quality	• Monitoring activities taking too long time to complete. There is probably technical debt. • Monitoring technical debt through static analysis tools	(+) Continuous monitoring enabled by the shared codebase practice. Enables improving actions as Dojos[Sato *et al.*, 2008] and technical presentations to leverage team members' knowledge (+) Static Analysis metrics serve as a threshold that drives actions when code quality decreases. (-) Static Analysis not necessarily improves code quality, but help to prevent quality degradation.
		Root cause analysis (RCA) of defects	(+) Focused on quality improvement, it also helps on process efficiency by identifying causes of waste (e.g.: rework needed to remove a defect), confirming the need for quality verification steps. (-) Reactive metric that shows problems that already happened.
Personnel	Teamwork	Self-managed teamwork assessment [Campion *et al.*, 1993]	(+) Generate interesting insights and can be used in a regular basis. (-) Generates some overhead on data collection and discussion.

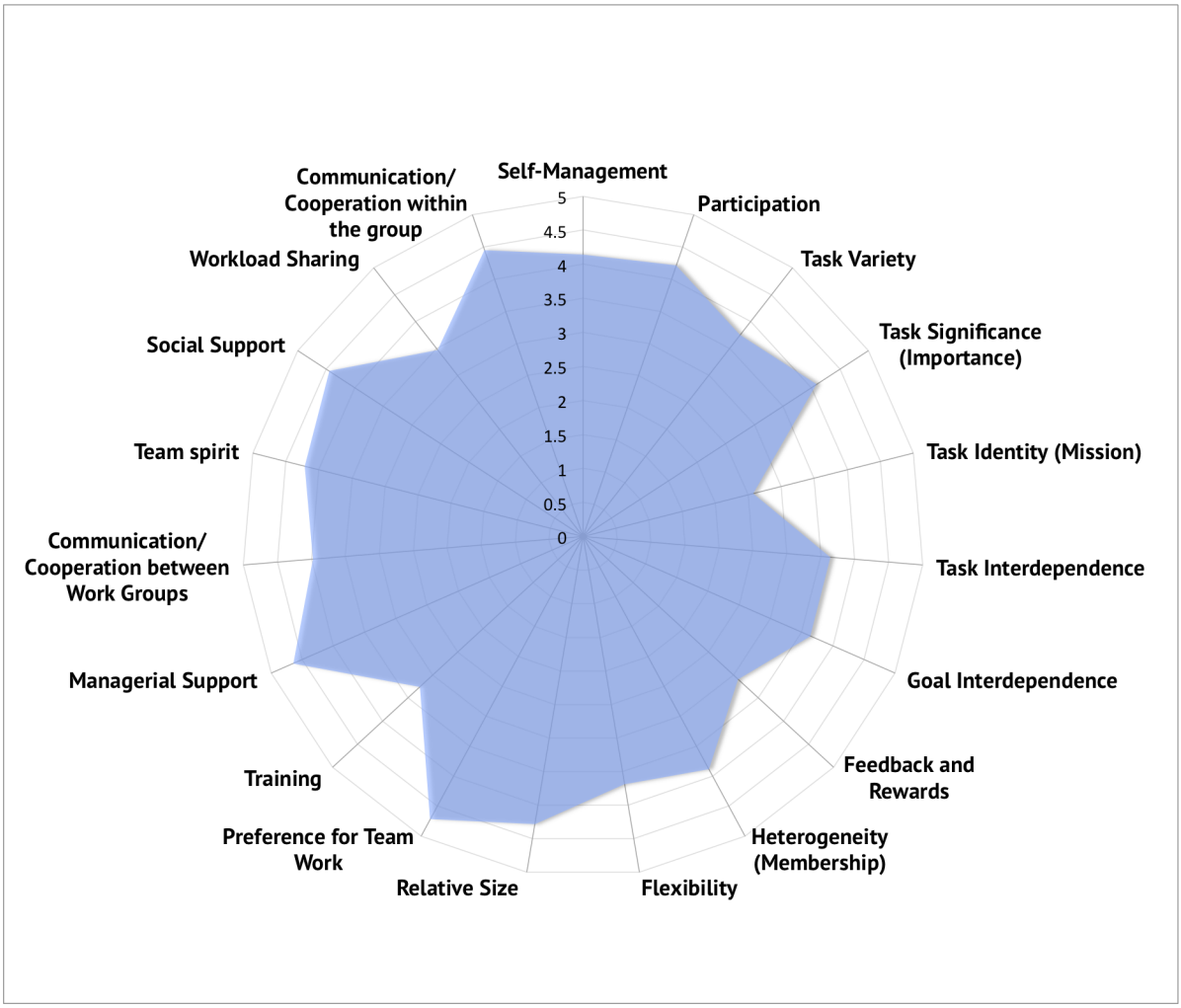

Figure 7.9: *Teamwork assessment results, based on Campion et al. [1993] protocol*

agreement. They asked to repeat the assessment and discuss results in the shortest possible time after data collection.

When asked about learning from technical debt monitoring, a developer said:

> "Software development requires people able to understand code quality implications. If you develop just thinking of now, you're gonna pay it in the next features [...] This short-term economy will harm the product in 5-6 months. Similar features will require the same time in the feature." *(Developer 2)*

This reflection generated also actions. The developer decided to talk to the client, to make the technical debt clear and confirm whether the debt was consciously acquired. The client did not know about the debt. The team and the client agreed to organize Dojos [Sato *et al.*, 2008] as a safe space for learning, in which developers could share coding practices. It was also important to set the correct expectation in some developers that, overall, team members' level of expertise on software quality is not the same. Some developers also adjusted their behavior to produce even better code as a way to avoid quality degradation when other (less prepared) developers work.

7.3 Eliciting research results

We, first, propose a simple instrument to guide agile team productivity monitoring through quantitative and qualitative metrics (Key finding 9). Then we propose an approach to tailor agile

team productivity monitoring to specific contexts within information systems and software engineering. Finally, we present reflections on monitoring usefulness for the team (Key finding 10).

7.3.1 Key finding 9: Agile team monitoring approach

The proposed approach to monitor agile team productivity is based on a simple instrument that aggregates productivity factors into five *dimensions*: *Product*, *Personnel*, *Project*, *Process*, and *Organizational*, the same dimensions we used to classify productivity factors in previous chapters. Factors represent monitoring *goals*. Teams need to identify these goals, the main drivers in their context, which might change over time.

By identifying key goals, the next step is to identify (or develop) quantitative and/or qualitative measures through research, establishing *how to monitor* the productivity goals. In addition, they need to implement the measures, which may require *actions*, as workshops, meetings, etc.

Finally, as part of the monitoring approach, teams must evaluate monitoring usefulness to achieve the assigned goals. Reflection is an important step that enables the necessary feedback cycle that enables productivity monitoring adaptability. This is why we added *evaluation* to the monitoring instrument.

Figure 7.10 illustrates the instrument we used throughout the project. It has the same structure we used to evaluate experiences and specify learning of action research cycles. It also resembles the well-known approach for defining measurement goals in software engineering, the Goal-Question-Metric (GQM) [Basili and Weiss, 1984], with a greater emphasis in the learning system behind the measurement (actions and evaluation).

Dimension	Goal	How to monitor	Actions	Evaluation
Product	[Quality, Innovation, etc.]			
Personnel	[Teamwork, Trust, Motivation etc.]			
Project	[Speed, Scope etc.]			
Process	[Leanness, Efficiency, etc.]			
Organizational	[Inter-team coordination etc			

Figure 7.10: *Proposed productivity monitoring instrument*

The action research also led to a framework for developing agile team productivity monitoring approaches. We suggest that a process similar to the one we followed (see Figure 7.11) can be used in other contexts within software engineering, particularly agile software development. It mixes **action research activities** (theoretical perspective) described in Figure 7.1, and **practical activities** we described in Chapter 3 on developing performance measurement systems (Figure 3.3).

The important underlying concept in our monitoring approach is its fitness to the agile development cycles. To be able to generate the intended value, the monitoring must continuously evolve and adapt to the current productivity context. Depending on the dimensions and the existing tensions surrounding the team, the monitoring goals will change. The monitoring approach also needs to consider the social system nature of a team

Because the monitoring approach combines both practical and theoretical perspectives, practitioners and researchers who want to replicate this case can adopt it.

7.3.2 Key finding 10: Monitoring usefulness

We discuss overall monitoring and measurement usefulness for the studied team based on our interviews. Our findings highlight some limitations in our approach, particularly on introducing

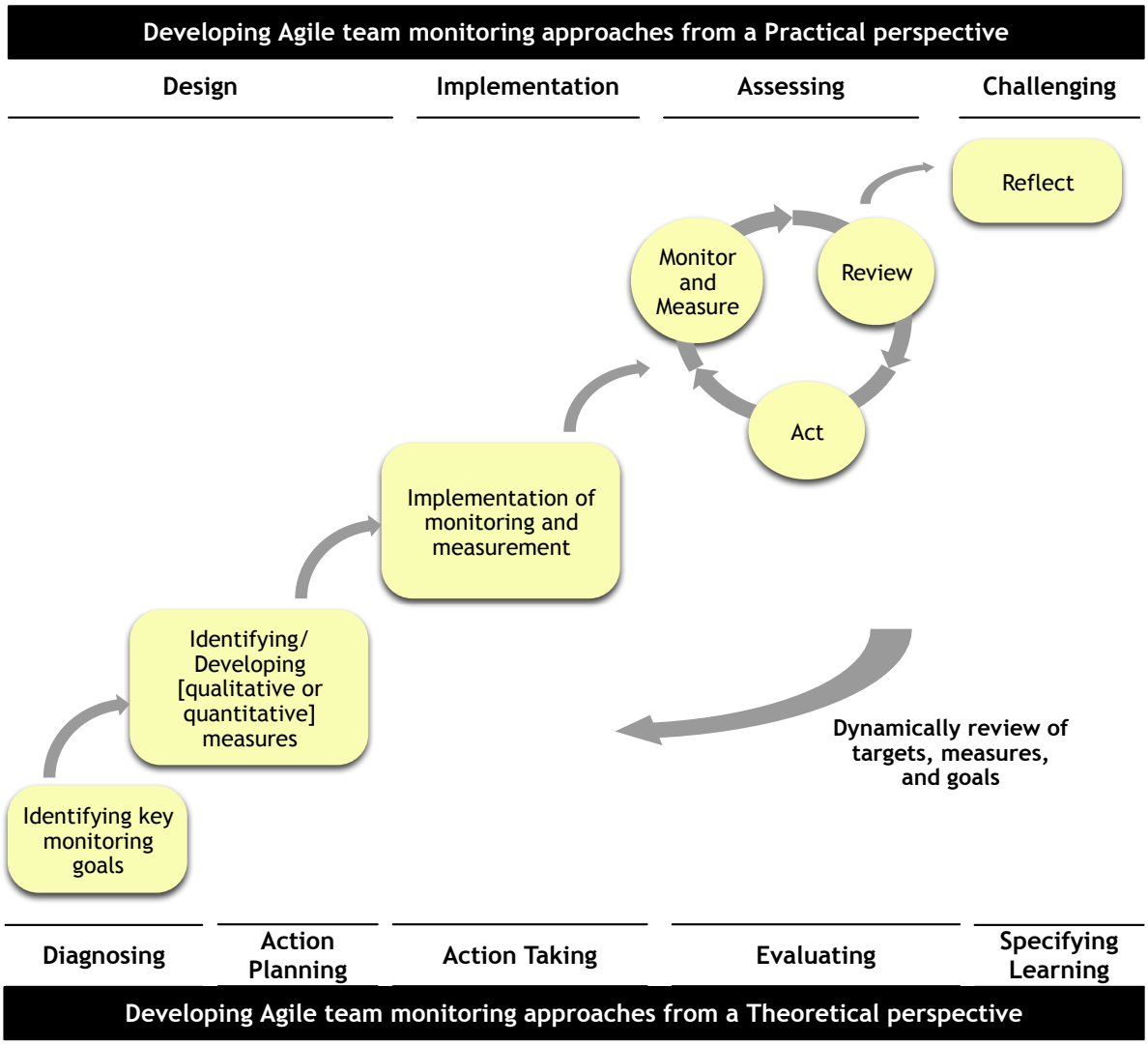

Figure 7.11: *Developing productivity monitoring approaches from Theoretical and Practical Perspectives*

metrics, and driving learning and change.

Introducing and removing metrics in the project

According to the team members, even metrics being incrementally introduced or removed in the project, which shows adaptability, it was not always clear *why* or *when* some metrics were introduced. This interferes with metrics goal setting and, consequently, their usefulness for the team. The communication and discussion of metrics in the team and in the organization is still incipient, even with the acknowledged presence of metrics experts in the team.

Metrics overhead

As the team has some people working with metrics, usually management, the team members do not feel that project metrics generates overhead. Teamwork assessment was the only metric that seems to generate some overhead. Despite this problem, they found the assessment worthy.

Productivity metrics might help just some groups of team members

Some metrics are meaningful for some groups of team members. For instance, the *Root cause analysis* practice was experimented by the team after one of the sessions to create a common

understanding of the work process. However, after some weeks, most team members were not aware that the experimentation was going on. At the end, the quality assurance analyst responsible for the experiment scheduled a session to share results just with other quality analysts. One could argue that an alternative explanation would be a specialization of roles. Nevertheless, this team regularly schedules sessions to share knowledge and discuss issues.

Metrics Usefulness

Interviewed team members agree that metrics data about the project help the organization to learn something useful. They also agree that their participation in such metrics adoption helps the organization to be more successful, which is the organizational usefulness of metrics. The team members felt they could make a change in the project metrics if they really felt the need and if the change was justified, and that is voluntary.

Metrics driving learning and change

Learning in the group level has been defined as "an ongoing process of reflection, and action, characterized by asking questions, seeking feedback, experimenting, reflecting on results, and discussing errors or unexpected outcomes of actions" [Stagl *et al.*, 2006]. Learning contributes to adaptation by using accumulated knowledge to scan changes, assign meaning, and adjusting [Stagl *et al.*, 2006].

Our observations witnessed some episodes of learning through productivity monitoring and measurement. Some metrics helped the team members to identify sources of waste, to create a shared understanding of productivity issues, and to highlight the importance of feedback. However, learning from metrics is not necessarily happening for everyone. A business analyst mentioned a potential negative effect from metrics when they condition people instead of fostering critical thinking:

> "Metrics teach for good and for evil. It's even worse when they condition people, as many other companies usually do to make them repeatable in other projects. As this project is the first agile project for many people here, maybe they think that this [what we are doing] is pure and right agile, you know? We need to show them this difference." *(Senior Business analyst 1)*

A developer describes a lack of understanding of the reasons to use metrics:

> "Sometimes people use the metric just to follow a rule [...] the lack of senior people might have influenced negatively [to understand metrics]. We need some experienced people to try to persuade us to use metrics [...] explaining 'this metric is important **because of**..." (Developer 1)

We thus interpret that, sometimes, people need guidance to enable members learning. A facilitator would promote discussions on adopted metrics and, more importantly, encourage critical thinking about why the team needs them.

7.4 Discussion

Through a 10-month action research study in a multinational company, we investigated how to monitor agile team productivity and usefulness of some metrics. Our action research brought skilled professionals from industry and academia with the strengths of each community to solve industrial research needs.

We now discuss our results in light of our propositions presented in Chapter 3, since we are proposing an approach to monitor agile team productivity.

Proposition 1: Productivity measurement should be used predominantly for learning, short-term management, and long-term improvement. As it is a dynamic measure by nature, it may differ with the products and the technology used. The set of metrics may vary at different stages of a product life cycle and should be developed in a process-oriented, bottom-up approach. Our findings pointed out that productivity measurement used for control, as the use of velocity in Cycle 0, leads to low performance as demotivation and lack of trust. When measurement was used as a tool to verify hypothesis or bring insights, the team members stopped to complain about metrics drawbacks. Different metrics were adopted throughout the project, since the team was adapting to context changes. Metrics evolved organically, and all team members could propose monitoring and measurement approaches. Overall, there was a productivity improvement from a team perspective - turnover was reduced, and motivation and trust somewhat strengthened.

Our findings confirm previous research that suggest that the ultimate goal of performance measurement should be *learning*, *improvement*, and *goal achievement* rather than control, driven by employees [Bititci *et al.*, 2011, Brudan, 2010, Davenport and Harris, 2007]. This research also adds to literature by providing a detailed experience on how measurement for learning occurs in an agile team.

Proposition 2: Software development productivity should be evaluated in a holistic way, since it involves different perspectives, human knowledge, learning, creativity, as well as performance and profitability. Our approach applied different perspectives of software productivity by evaluating timeliness, product quality, absenteeism, process efficiency, and teamwork. Due to time constraints, it was not possible to verify all productivity dimensions proposed by Drucker [1994] and Ramírez and Nembhard [2004], as well as their usefulness for agile productivity monitoring.

Proposition 3: It is possible to develop a Performance Measurement System within agile teams without generating additional overhead, since metrics can be chosen, integrated, and analyzed incrementally, as it is normally done in an agile project. Our proposed framework to monitor and measure agile team productivity (Figure 7.11) has roots on the performance measurement system phases described by Bourne *et al.* [2000]. Metrics were developed/chosen and analyzed in an incremental fashion through the research cycles. However, we found evidence that the process of introducing new metrics was not clear for all team members, and that metrics may generate overhead. Our findings thus partially support this proposition.

Proposition 4: Monitoring team performance is a team action to reach adaptive capabilities. Measuring is part of a continuous productivity monitoring for self-managed adaptive teams. Proposition 5: Agile productivity monitoring processes are important action processes to develop teams towards adaptive capabilities - a fundamental attribute for agile teams. In this study, we did not evaluate the team adaptive capabilities. Despite productivity monitoring had triggered adaptive actions, we did not address how the team adaptive capabilities for self-management evolved. Notwithstanding, we consider this study a first step for future investigations on continuous productivity monitoring for self-managed adaptive teams.

Proposition 6: Developing a process to monitor productivity factors must take into account agility variables, such as flexibility, learning, and responsiveness. Our proposed framework to monitor agile team productivity embodies a continuous evaluation of targets, measures, and goals to support flexibility and responsiveness. Learning takes place by challenging monitoring goals results through reflection (Figure 7.11). Our results show three cycles in which monitoring and measurement goals were set from distinct product, personnel, and process drivers. These findings cast light on the issue of monitoring productivity in a highly adaptive environment.

To the best of our knowledge, there was no similar evidence in the literature on how productivity is monitored in such flexible way.

Proposition 7: Agile team productivity factors should be investigated in light of a representative conceptual framework based on the nature of agile teams, such as self-organization, teamwork, and autonomy. The use of our conceptual framework not only supported the multiple-case analysis in Chapter 6, but also helped us to better delineate the company and team productivity factors. Through the conceptual framework lenses, we collected and interpreted data considering self-organization and teamwork as important context modelers. On the one hand, it biases our results, because we were focused on such nature of agile teams. On the other hand, it helped us to understand the team mindset, values, and beliefs.

7.4.1 Evaluating the Participatory Action Research

In their book *Virtuosity*, Larsen and Larsen [2012] explain learning from two perspectives:

> "Take History, for example. Let's say Diana wants to become the world's foremost expert on 13th century northern Italian village community life. Diana1 reads everything that's been written and gains an ability to repeat it to others. Some folks would call that learning. We think she has a shadow of understanding [...] Diana2 travels to northern Italy to live in a small village; researches 13th century recipes, seeks out ingredients, and cooks for her friends; uses technology of the time to sew period clothing that she wears to see how it feels; makes reproductions of tools of the era and uses them in daily life; hunts the language from elder speakers so she can have conversations with descendants of those who lived at the time; and so on. Diana2 becomes a history hunter, so she can apply the insights of 13th century northern Italian village life, see hidden connections, empathize and understand the era deeply."

From a researcher perspective, this metaphor represents the difference between case studies and participatory action research. Comparing with our previous studies, this engagement offered a deep and live experience on how to apply empirical knowledge in a real context. Overall, participatory action research was an effective research method to solve complex adaptive issues such as agile team productivity.

The following discussion validates the research process by clarifying the roles played by the members involved, reviewing the approach of data collection, and explaining how the process was controlled. Each of these issues played a key role in ensuring research relevance and rigor.

Roles. To establish action research credibility, changes in the social setting are analyzed. For instance, it is expected that research participant roles will noticeably change during the project. Our research was conducted entirely on-site. In all other activities, the role of practitioners was to provide context and an environment for the monitoring approach. I was part of the local research group and, at the same time, part of the management team. This helped us to observe and gain context, but at the same time might have biased the interpretation. We overcame this limitation by triangulating multiple data sources.

Documentation. Credibility is established through triangulation of information from multiple data sources: data collection included transcribed interviews, official and unofficial company documents, e-mails, observation notes, and planning session results. Credibility also relies on the rigor with which the context and special nature of the research scene is described and analyzed.

Action research credibility involves also checking researcher interpretations, theories, and tentative conclusions with others before, during, and after the research process. There were many informal group sessions in which we had the opportunity to share thoughts and interpretations

with the team. By informal we mean that, most of the time, it was not clear for the team members that I was playing a researcher role, instead of the usual member role.

Control. Action research projects can be classified as *formal*, *informal*, and *evolved* according to the control structures adopted [Avison *et al.*, 2001]. Informal control structures might begin and finish without formal letters (at most, broad and general written agreements). Our research project was informally controlled because it did not involve any specific non-disclosure agreement, except for that signed by the employee (myself). This informal control structure does not prescribe mechanisms of researcher engagement into the organization, not even the collaborative team composition, or warrants for action.

From the domination viewpoint, this research fits into the *identity domination*, which means that researchers and practicing organization professionals are the same person (or persons) [Avison *et al.*, 2001]. One or more of the researchers are internal members of the practitioner organization, and already possessed the action warrant authority necessary to make the interventions.

Usefulness. The validity of an action research contribution depends intrinsically on whether some desirable change was created [Iversen *et al.*, 2004]. Actions, outcomes, and therefore, changes were acknowledged by company D's employees, as the team members, employees working in projects with the same client, and directors.

Changes were introduced throughout the project to test their effect and usefulness (e.g., actions to improve trust and motivation problems). We also used results from each stage in later phases.

We agree with Sandberg *et al.* [2011] when they argue that action research (collaborative or, in our case, participatory) takes time for both parties in the process: researchers and industry. Sandberg *et al.* [2011] say "researchers must learn to be agile toward industry needs, and practitioners must learn to appreciate research rigor. Such learning requires time, perseverance, and conscious reflection".

Theory. The monitoring framework represents an advancement of state-of-the-art in agile team productivity. We clarified the relationship between existing theory and our results by discussing how our study findings support the propositions from literature (Chapter 3).

Transfer. The purpose of this research is to expand the understanding of an issue that our findings show relevant: agile team productivity (Chapter 5). We provided details on the changes that occurred and related to general concepts from the literature to explain their nature and underlying causes. The suggested framework is presented as a generic approach rather than a procedure to be followed. Using the approach requires a facilitator with experience with productivity factors, agile team dynamics, and a general competence in organizing workshops and required sessions to discuss monitoring outcomes.

7.5 Summary

This chapter presented our results on agile productivity monitoring and measurement based on a 10-month action research, addressing RQ3. Our main contribution was to provide **C5. A team productivity monitoring process considering agility and an evaluation of the usefulness of agile team productivity metrics**.

Chapter 8

Conclusion

This thesis contains the results of several empirical studies performed involving three Brazilian companies, one multinational company, and a survey with Brazilian IT community. These studies combined literature study, collecting and analyzing quantitative data from a national survey, and mixed data sources from three case studies and an action research study. A mixed-method research design was applied to allow taking benefit of all available data, combining the results and answering questions that are not possible to answer otherwise.

The research problem addressed in this thesis was to *explore software productivity in an up-to-date view of software development*. The problem was narrowed in three research questions:

- **RQ1. How important is productivity for companies adopting agile methods? How do they define productivity?**
- **RQ2: What factors impact agile team productivity and how is this impact from the team point of view? Which agile practices are perceived to impact on a given team's productivity?**
- **RQ3: How to monitor productivity factors, considering agility and adaptability? How do agile metrics support productivity monitoring?**

Through a synthesis of contributions from nine papers, two book chapters, and two industrial reports, we have identified ten key findings connected to the three research questions. The contributions of this thesis build on the results and findings of the research studies presented on Chapters 5, 6 and 7.

8.1 Summary of results

Empirical research is performed to verify theories, develop new theories or extend existing ones, and improve the practice. The thesis contributes to these aspects by:

Describing multiple aspects of software productivity. While agile methods, as XP and Scrum, promise increased productivity, the agility concepts seems to focus in opposite ideas, such as adaptability and flexibility. In addition, a large number of studies still take a simplistic view of productivity such as measuring lines of code/hour. The contribution of this thesis (described in C2) is the overall synthesis of the multiple aspects involved in productivity, putting software development as knowledge work, and explaining that productivity is a constituent part of agility, instead of being in a paradoxical relationship.

Verifying the importance of productivity for agile teams in the Brazilian IT context. Adding to the previous contribution, and answering RQ1, this thesis not only argue for the

importance of productivity for agile teams, but also verified it through a survey in the Brazilian IT industry. 471 responses were collected over 6 months, showing that the most important reason for adopting agile methods was productivity increase.

The results of the correlation analysis showed that the perceived productivity benefits from agile methods adoption are not correlated with company size or experience. Inasmuch as there is no similar evidence on past research to contrast our results, this finding opens a new direction for further research on agile benefits.

Our survey analysis also showed that companies that perceived a high productivity improvement from implementing agile methods usually adopt the same agile practices that most respondents adopt.

Verifying that soft factors are prominent issues influencing agile team productivity. Based on the answers to RQ2 and RQ3, agile teams must be aware of the magnitude of soft productivity factors. Team design, team member turnover, members' motivation, and trust are outstanding influences that makes an important contribution for team success. Soft factors are hard to measure, practitioners need to adopt qualitative strategies to recognize them.

The contributions described in C3, C3.1, and C5 are the essential contribution of this thesis in this respect.

Generating new theories, hypotheses, and methods. Productivity factors should be studied in combination with one another, since they have multiple and crosscutting impacts on team productivity. For example, we studied different companies with similar factors, but different contexts to better understand underlying reasons for factors.

The major original contributions of our research on factors are (1) to explore, in an industrial setting, a significant agile productivity factor: *team management*, (2) to analyze the mechanisms underlying team management using a novel conceptual framework, and (3) to detail the conceptual framework by adding links among its components, enabling further tests through confirmatory studies. We give new insights on possible cause-effect relationships between staff turnover, team design choices, and inter-team coordination factors and the observed productivity outcomes. The framework is ready for testing through confirmatory studies, contribution described in C4.

From a methodological standpoint, we integrated positivist (survey), interpretative (multiple-case studies), and pragmatic (action research) approaches. Pragmatists use any available methods and strongly prefer mixed methods research, where several methods are used to shed light on the issue under study. We claim that research in software development is a social construction that requires a multi-method approach. This thesis contributes to the literature by describing this integration that can be applied in other empirical software engineering studies.

Proposing an agile team productivity monitoring. There are multiple possible productivity factors influencing a single agile team. In addition, factors interact with each other, bounded by context, and they often change. Productivity is a dynamic measure by nature, therefore it is not possible, or even useful, to build a single productivity monitoring system for an agile team.

Considering context changes, productivity monitoring must focus on short-term goals, which are related to five well-known dimensions: product, personnel, project, process, and organization. Each dimension drives different goals throughout time, demanding team agility. We provided a simple framework to help teams define goals linked to the five dimensions, and some examples from the action research study, as described in contribution C5.

Considering the effects of applying the method in practice, it seems we succeeded in helping agile teams to better monitor their productivity in an adaptive context. Our results also

showed advantages of adopting more than one measurement approach to monitor productivity. Productivity is a multifaceted concept, and neglecting this fact may lead to a dysfunctional measurement that hides the true team performance.

Verifying the usefulness of metrics for agile team productivity monitoring. Measurement can be used to monitor agile team productivity when learning is the main goal. However, teams should not expect that all metrics would foster genuine double-loop learning. They need to consider strategies to introduce and remove metrics, metrics overhead, and a presence of a facilitator to promote discussions that encourage reflection and critical thinking.

8.2 Future Research

Based on the experiences with productivity factors of agile teams, we see several possibilities for further improvement of the method. We suggest:

The role of adaptability on team productivity factors. Context changes and other demands for adaptability influence a great deal on team productivity. More research to investigate individual, team, and organizational adaptability characteristics that foster or hamper team productivity is needed. Such studies should also evaluate what characterizes an adaptation moment for a team.

Metrics driving learning and change. Detailed studies on how metrics support double-loop learning and change are needed. Learning and change are preconditions for team adaptation. Future studies should focus on how to identify learning episodes and their types (single-, double- or triple-loop learning). Associating learning episodes from metrics with changes may shed light into this question. Researchers should also investigate differences between individual and group learning, the latter is the ultimate goal for a team.

Productivity drivers in agile companies. An exploratory factor analysis in our survey data can be done to investigate correlations between productivity and other variables such as time to market, quality, cost, etc. Such exploration might reveal whether respondents interpret productivity similarly to these other variables or not. This study would generate contemporary quantitative evidence on how companies are interpreting productivity in large scale.

Appendix A

Survey protocol

A.1 Research Problem

Despite the fact that agile methods have been increasingly adopted and have rapidly joined the mainstream of development approaches, their adoption in the Brazilian IT industry has not been studied much in the literature. We aim to investigate the inception, growth and establishment of agile methods in this community.

A.1.1 Research questions

We aim to investigate the inception, growth, and establishment of agile methods in this community through *RQ3: How agile methods practice has evolved in the Brazilian IT industry?*.

Our main goal was to take an initial step towards understanding the agile methods state-of-the-practice in the Brazilian IT industry. To better focus our data collection and analysis, we divided RQ3 into six sub-questions:

- SQ1. What are the characteristics of agile practitioners?
- SQ2. What are the characteristics of agile companies?
- SQ3. What were the companies contexts when they adopted agile?
- SQ4. How agile methods are being adopted?
- SQ5. What are the perceptions after agile adoption?
- SQ6. What are the main challenges when adopting agile?

The relationship between the constructs we want to investigate and the sub-questions is described in Table A.2.

A.1.2 Research design

The research phases are:

1. Research protocol development;
2. Survey data collection;
3. Survey analysis (descriptive and maybe inferential);
4. Selection of interesting findings from survey analysis (researchers will discuss and select)

A.1.3 Research instrument

Table A.1: *Sub-questions and Constructs*

Research sub-questions	**Construct**
What are the characteristics of agile practitioners?	Personal Profile (agile experience, position, current use)
What are the characteristics of agile companies?	Company Profile (IT department size, Business domain, Location, agile experience)
What were the companies' context when they adopted agile?	Company context (reasons, worries, champion)
How agile methods are adopted?	Agile usage (method adopted, practices adopted, %projects using, distributed)
What are the perceptions after agile adoption?	Perceptions of usage (speed, major benefits)
What are the main challenges when adopting agile?	Challenges (agile projects failure, barriers for further adoption)

Table A.2: *Sub-questions and Constructs*

Construct	**Survey question**
Personal Profile> current use agile development?	What situation below best describes your current level of exposure to I've heard about it, starting to study it more Considering to adopt agile methods for the first time Member of an agile team in the past Agile coach Agile team leader Member of an agile team
Personal Profile>Position company?	Which role below best describes your current position in your Product Manager President/CEO CIO/CTO Consultant QA/Tester IT manager Architect Project manager Other Team lead Senior developer Developer
Personal Profile>Agile experience methods?	How long have you personally been practicing agile development Never Less than 6 months 6-12 Months 1-2 Years 2-5 Years > 5 Years

Company Profile>IT department size	How large is your total software organization? 1 - 5 people 6-20 people 21-50 people 51-100 101-250 > 250
Company Profile>Business area	What is the main activity of your organization? Multimedia ERP Games/Entertainment Embedded systems Mobile Energy Health care Financial Consulting/Services Communications Scientific/Engineering Education Other Escritório Governo Internet
Company Profile>Localization	Where your company is located?
Agile usage> %projects using agile method?	What percent (%) of your company's software projects use an agile
Agile usage> Distributed teams	Do you work in a company with distributed development teams?
Company Profile> agile experience	How many years has your company been practicing agile? Never Less than 6 monts 6-11 Months 1-2 Years 2-5 Years > 5 Years
Company context>reasons	What were the reasons for adopting agile within your team or organization? Enhance ability to manage changing priorities Increase productivity Enhance software quality Improve alignment between IT and business Improve project visibility Accelerate time-to-market Simplify development process Improve Engineering Discipline Reduce cost Enhance software maintainability/extensibility Improve team morale Reduce risk

Perceptions of usage>speed	What is your perception regarding adoption of agile in your company/team?
	Faster time to completion
	Not yet completed and agile project
	Same time to completion
	Not yet completed an agile project
	Slower time to completion
Company Context> worries	What were the greatest concerns in your team or organization when adopting agile methods?
	Inability to scale
	No worries
	Reduced software quality
	Regulatory compliance
	Lack of engineering discipline
	Development team opposed to change
	Management opposed to change
	Quality of engineering team
	Loss of management control
	Lack of upfront planning
	Lack of predictability
	Lack of documentation
Company Context>champion	What is/was the champion of agile adoption in your company?
	Team leader
	Developer
	Project manager
	IT manager
	Director of Development
	CIO/CTO
	CEO
Agile usage>method	What is the most followed agile method in your company?
	Scrum
	Extreme Programming (XP)
	Scrum/XP Hybrid
	Lean Development
	Scrumban
	Dynamic Systems Development Method (DSDM)
	Agile Unified Process (AgileUP)
	Agile Modeling
	Feature-Driven Development (FDD)
	Customized
	Don't know
	N/A
Challenges>barriers	What are the barriers to further adoption of agile in your current organization?
	Ability to change organizational culture
	General resistance to change
	Availability of personnel with necessary skills
	Management support
	Project complexity or size
	Customer collaboration
	Perceived time to transition

	Budget constraints
	Confidence in ability to scale agile
	Other
Agile usage> practices	What are the agile practices adopted in your company?
	Other
	Unit testing
	Daily Standup
	Release planning
	Continuous integration
	Automated builds
	Burndown
	Retrospectives
	Refactoring
	Continuous deployment
	Coding standards
	Velocity
	Test Driven Development (TDD)
	Automated acceptance testing
	Kanban
	Colletive code ownership
	Digital taskboard
	Open workareas
	On-site customer
	Pair programming
	Behavior Driven Development (BDD)
	Iteration planning
Challenges> agile projects failure	What were the main reasons for failed agile projects (if
any)?	
	Lack of cultural transition
	Lack of management support
	Unwillingness of the team
	Insufficient training
	Other
	External pressure to follow traditional waterfall practices
	Company philosophy/culture at odds with core agile values
	Lack of experience with agile methods
	Did not happen in my organization
Perceptions of usage> Major Benefits	What value have you actually realized from implementing
Agile practices?	
	Manage distributed teams
	Cost reduction
	Engineering discipline
	Reduce risk
	Business/IT alignment
	Maintainability and extensibility
	Time-to-market
	Simplify development process
	Software quality

	Productivity Project Visibility Team morale Enhance ability to manage changing priorities

Appendix B

Case studies protocol

B.1 Instrument 1: Company Characterization

1. The industry in which products are used (e.g., banking, consumer goods, telecommunications, travel): 2. Nature of the software development organization: 3. The skills and experience of software staff (See Instrument 2) a. with a language, a method, a tool, an application domain. 4. The type of software products used (e.g., a design tool, a compiler) : 5. The software processes being used (e.g., a company standard process, the quality assurance procedures, the configuration management process): 6. Number of employees: 7. Number of IT employees: 8. Organization context – another details (free space to write about organizational context):

B.2 Instrument 2: Characterization of the Project and the Team – Interviews and Questionnaire

Table B.1: *Characterization of the Project and the Team*

Team Characterization	Project <name>
Team size	
Specific Roles in the team (developer, UX, PO, tester etc)	
System short description	
Language	
Platform (mainframe, mid range, multi-platform, or PC)	
Software complexity (level of complexity of the application in development)	

B.3 Instrument 3: Interview guide

- What is your role in the project and how long have you been working on it?
- How is the team's 'way of work'? What is your role in the team?
- How does the prioritization of feature work?

Table B.2: *Agile practices adopted by team*

Practice	Usage (Full/Partial/Don't use)	Comments
Planning game		
Retrospectives		
Shared Code		
Code & Tests		
Single code base		
Real customer involvement		
Root cause analysis		
Daily deployment		
Pay-per-use		
Negotiated scope contract		
Energized work		
Whole team		
Ten minute build		
Continuous integration		
Weekly cycle		
Monthly cycle		
Stories		
Incremental design		
Other <name>		

- How does the team judge, during prioritization, the value to be delivered?
- In your opinion, has the customer realized the delivered value in each iteration and release? Does he report this?
- In your opinion, has the team delivered value to the customer?
- What is the project status (scope, cost, time box, etc.)?
- What is your opinion regarding the project productivity? How do you perceive this productivity?
- What is your opinion about project quality? How do you track the external quality? And internal? How do you handle the bugs?
- Is there any re-work on the project? How much?
- What do you think that most influences your team's productivity?
- In your opinion, which changes were recently done in the team way of work that can have influenced any productivity variation?
- Do you consider the project motivating?
- Is there anything demotivating you in the project? And in the company?
- Is there anything in the agile methods that motivates you in anyway?
- Is there any kind of waste that jeopardizes the project?

- If you could choose three things to increase productivity, what would them be?
- Do you think that the use of agile methods increases team productivity? Why? Is there something in Agile that helps your own productivity or the productivity of your team?
- Is there anything in the agile methods that decreases your individual productivity or the team productivity? If so, what is it? Why?

B.4 Instrument 4: Observational protocol - Questions and frequency

- Is there some mention about events that are affecting the team productivity during the daily meeting? (Daily)
- Is there some mention about events that are affecting the team productivity during the retrospective? (Per iteration)
- Is there some mention about events that are affecting the productivity during conversations with team? (Daily)
- What is the suitability of the workplace to do creative work, e.g., windows, natural light, size of room and desk? (Per iteration)
- What are the ways in which all actors interact and behave toward each other? (Daily)
- Was there anything unexpected? (Daily)

B.5 Instrument 5. Productivity Measurement protocol

Collect the productivity at the end of iteration.

Table B.3: *Characterization of the Project and the Team*

	Iteration	Story points/hours	Variation (%)
#1	13/Sep/2010 – 28/Sep/2010	X	–
#2	01/Oct/2010 – 14/Oct/2010	Y	k

Appendix C

Thematic analysis coding evidence

C.1 Code Analysis

We adopted different types of code analysis to identify themes. One of the techniques was visualization of codes, in which color and size represents the code frequency in its group.

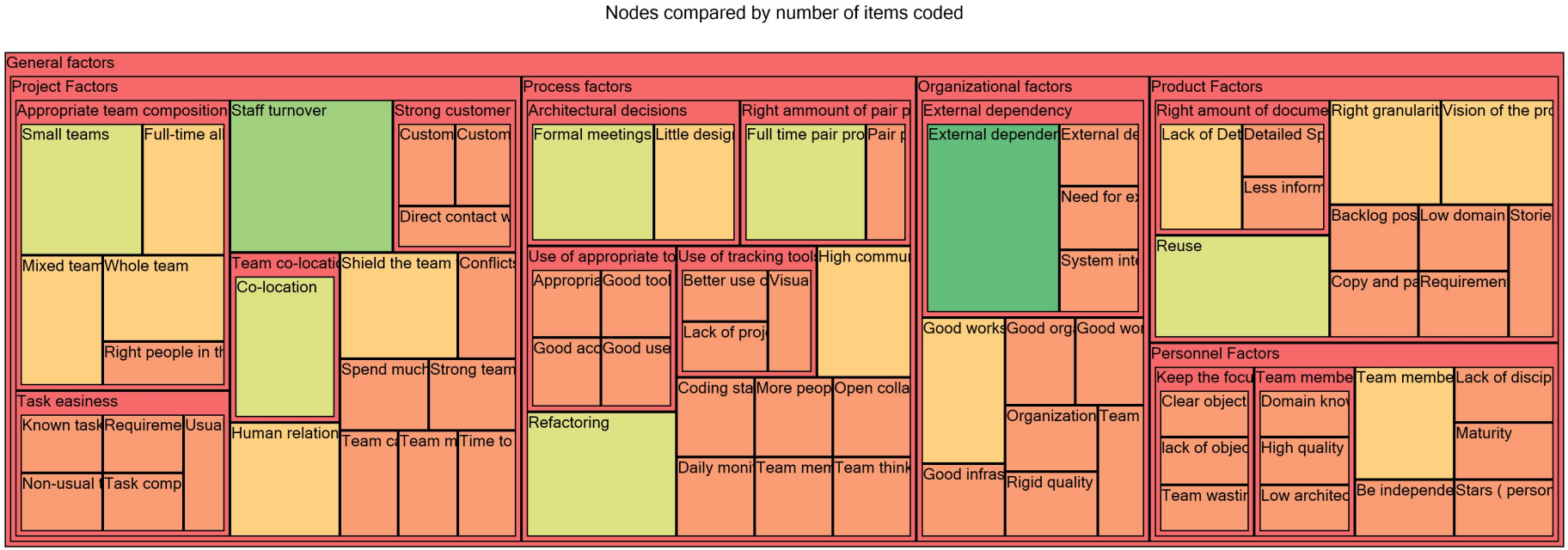

Figure C.1: *Code analysis using visualization techniques*

C.2 Excerpt from a code list report

Name	Memo Link	Sources	References	Created On	Created By	Modified On	Modified By	
Agile practices nodes		16	42	11/2/2011 13:04	COM	11/2/2011 13:13	COM	
	Acceptance tests as documentation		1	1	18/2/2011 11:52	COM	20/2/2011 23:03	COM
	Agile tayloring		1	1	18/2/2011 14:15	COM	20/2/2011 23:06	COM
	Automated tests		1	1	18/2/2011 11:54	COM	20/2/2011 23:00	COM
	Coding standard		1	1	11/2/2011 15:12	COM	20/2/2011 19:17	COM
	Co-location		3	3	11/2/2011 15:25	COM	16/3/2011 13:47	COM
	Constant feedback		2	2	11/2/2011 15:23	COM	16/3/2011 14:18	COM
	Customer involvement		2	2	17/2/2011 20:08	COM	20/2/2011 20:55	COM
	Flexible scope can lead to procrastination		2	2	18/2/2011 13:07	COM	18/2/2011 14:18	COM
	High collaboration - as communities - Principle		1	1	17/2/2011 17:48	COM	20/2/2011 23:05	COM
	High commitment		2	2	18/2/2011 14:36	COM	20/2/2011 22:04	COM
	Incremental software development		2	2	18/2/2011 13:00	COM	16/3/2011 15:14	COM
		Short iterations		1	1	11/2/2011 15:23	COM	20/2/2011 22:48
	Individuals over processes		1	1	20/2/2011 20:45	COM	20/2/2011 20:51	COM
	Informative workspace		1	1	18/2/2011 15:23	COM	20/2/2011 22:46	COM
	Lack of sustainable pace		1	1	16/3/2011 3:43	COM	16/3/2011 3:43	COM
	Little documentation		3	3	17/2/2011 20:06	COM	16/3/2011 14:13	COM
	Pair programming		5	6	11/2/2011 15:13	COM	23/2/2011 20:09	COM
		Pair programming in easy tasks		2	2	11/2/2011 14:40	COM	23/2/2011 20:09
	Retrospectives		1	1	16/3/2011 3:28	COM	16/3/2011 3:32	COM
	Shared code		2	2	18/2/2011 14:49	COM	21/2/2011 1:00	COM
	Short and frequent meetings		3	4	16/3/2011 13:45	COM	16/3/2011 16:20	COM
		Daily meetings		1	2	18/2/2011 12:00	COM	20/2/2011 22:57
		Stand up meetings		1	1	17/2/2011 19:59	COM	20/2/2011 20:41
	Sprint planning		1	1	16/3/2011 3:32	COM	16/3/2011 3:32	COM
	Sprint review		1	1	16/3/2011 3:33	COM	16/3/2011 3:33	COM
	Team empowerment		1	1	16/3/2011 15:02	COM	16/3/2011 16:36	COM
	Too many meetings		1	1	16/3/2011 3:35	COM	16/3/2011 3:35	COM
	Whole team		1	1	11/2/2011 15:12	COM	20/2/2011 19:17	COM
General factors		21	151	11/2/2011 13:05	COM	21/2/2011 17:45	COM	
	Organizational factors		14	25	21/2/2011 11:57	COM	21/2/2011 13:32	COM
		Bonuses and incentives for motivating employees		2	2	16/3/2011 3:23	COM	16/3/2011 12:53
		Conflicts between teams		2	2	16/3/2011 13:37	COM	16/3/2011 16:31
			Physical space to enable communication between teams		1	1	16/3/2011 14:48	COM
		Events with external people to share knowledge		1	1	16/3/2011 14:49	COM	16/3/2011 14:49
		External dependency		8	11	18/2/2011 19:02	COM	21/2/2011 11:57
			External dependencies management and prioritization		2	2	18/2/2011 13:55	COM
			External dependency (general)		6	7	11/2/2011 13:15	COM
			Need for external definitions and customer explanation		1	1	11/2/2011 15:28	COM
			System integration		1	1	11/2/2011 13:55	COM
		Good infrastructure		1	1	11/2/2011 15:49	COM	21/2/2011 11:36
		Good organizational environment		1	1	18/2/2011 13:51	COM	21/2/2011 2:02
		Good workspace		3	3	11/2/2011 14:54	COM	16/3/2011 16:31
			Good workspace layout		1	1	18/2/2011 12:08	COM
		Organizational investment on quality		1	1	18/2/2011 15:21	COM	21/2/2011 12:12
		Rigid quality policies in the integration environment		1	1	11/2/2011 15:11	COM	21/2/2011 12:27
		Specific time to work in a personal project		1	1	16/3/2011 14:45	COM	16/3/2011 14:45
		Team spending time creating infrastructure		1	1	11/2/2011 14:19	COM	21/2/2011 12:48
	Personnel Factors		10	15	11/2/2011 16:14	COM	21/2/2011 13:34	COM
		Be independent		2	2	18/2/2011 11:25	COM	16/3/2011 2:57
		Keep the focus		3	3	21/2/2011 12:59	COM	21/2/2011 17:42
			Clear objectives in the meetings		1	1	11/2/2011 14:09	COM

		lack of objectivity can lead to difficulty in agreeing		1	1	18/2/2011 15:17	COM	
		Team wasting time discussing technical issues		1	1	17/2/2011 19:36	COM	
	Lack of discipline and routine		1	1	18/2/2011 13:21	COM	21/2/2011 11:49	
	Maturity		1	1	17/2/2011 17:09	COM	17/2/2011 17:09	
	Stars (person who knows a lot)		1	1	17/2/2011 17:14	COM	21/2/2011 19:39	
	Team members with competence and expertise		3	3	21/2/2011 13:03	COM	21/2/2011 17:43	
		Domain knowledge		1	1	18/2/2011 12:33	COM	
		High quality of technical team		1	1	17/2/2011 17:09	COM	
		Low architecture experience		1	1	11/2/2011 14:12	COM	
	Team members with great motivation		4	4	11/2/2011 15:20	COM	16/3/2011 15:28	
		Motivation		2	2	16/3/2011 3:05	COM	
Process factors		17	37	18/2/2011 17:44	COM	21/2/2011 13:37	COM	
	Architectural decisions		0	0	21/2/2011 13:12	COM	21/2/2011 13:39	
	Coding standard - Agile Practice		1	1	11/2/2011 15:02	COM	21/2/2011 1:20	
	Daily monitoring		2	2	11/2/2011 16:02	COM	16/3/2011 14:54	
	Formal meetings to discuss technical things		3	3	11/2/2011 15:08	COM	21/2/2011 16:23	
	High communication		3	3	11/2/2011 15:22	COM	16/3/2011 14:56	
	Little design up front		2	2	11/2/2011 15:37	COM	21/2/2011 19:45	
	More people involvement in all development phases		2	2	18/2/2011 12:55	COM	16/3/2011 14:56	
	Open collaboration		2	2	17/2/2011 17:20	COM	17/2/2011 17:30	
	Pair programming		6	6	21/2/2011 13:17	COM	16/3/2011 13:38	
		Full time pair programming		3	3	18/2/2011 11:46	COM	
		Pair programming helping on knowledge sharing		1	1	11/2/2011 13:47	COM	
	Refactoring		3	3	17/2/2011 16:43	COM	23/2/2011 20:05	
	Team members proximity		1	1	17/2/2011 20:23	COM	21/2/2011 12:45	
	Team participating on the feature definition		1	1	16/3/2011 3:19	COM	16/3/2011 3:19	
	Team thinks they are productive just because of the use of agile methods		1	1	17/2/2011 19:42	COM	17/2/2011 19:43	
	Team working		2	2	16/3/2011 3:15	COM	16/3/2011 17:05	
		Everybody can give opinions		1	1	16/3/2011 14:39	COM	
	Use of appropriate tools		3	4	21/2/2011 13:19	COM	21/2/2011 13:19	
		Appropriate tools		1	1	11/2/2011 15:50	COM	
		Good acceptance tool		1	1	11/2/2011 14:55	COM	
		Good tool to synchronize class models to database		1	1	11/2/2011 14:56	COM	
		Good use of automated tests		1	1	18/2/2011 14:41	COM	
	Use of tracking tools		3	4	21/2/2011 13:08	COM	21/2/2011 13:20	
		Better use of tracking tools		1	1	18/2/2011 12:31	COM	
		Lack of project tracking to learn with own mistakes		1	2	17/2/2011 19:46	COM	
		Visual tracking tool		1	1	18/2/2011 13:24	COM	
Product Factors		13	23	18/2/2011 17:44	COM	21/2/2011 13:38	COM	
	Backlog postponing		1	1	18/2/2011 11:32	COM	21/2/2011 1:11	
	Backlog reprioritization in the middle of the sprint		1	1	16/3/2011 13:05	COM	16/3/2011 13:05	
	Bugs in production		1	1	16/3/2011 14:09	COM	16/3/2011 14:09	
	Copy and paste errors (coding)		1	1	18/2/2011 11:24	COM	21/2/2011 1:24	
	Lack of product challenges		2	2	16/3/2011 3:08	COM	16/3/2011 13:31	
	Low domain complexity		1	1	11/2/2011 15:53	COM	21/2/2011 12:01	
	Prioritization considering only business concerns		1	1	16/3/2011 13:09	COM	16/3/2011 13:09	
	Requirement not clear		3	3	18/2/2011 11:39	COM	16/3/2011 14:05	
	Reuse		3	3	11/2/2011 14:42	COM	18/2/2011 12:44	
	Right amount of documentation		3	4	21/2/2011 11:56	COM	21/2/2011 11:56	
		Detailed Specification		1	1	11/2/2011 13:52	COM	
		Lack of Detailed Specification		2	2	11/2/2011 13:31	COM	
		Less informal requirements		1	1	11/2/2011 15:44	COM	
	Right granularity of stories		2	2	11/2/2011 13:59	COM	18/2/2011 14:34	

		Stories not really done		1		1	18/2/2011 12:39	COM	21/2/2011 12:29
		Vision of the product		2		2	17/2/2011 19:55	COM	21/2/2011 12:53
Project Factors			18	51		11/2/2011 16:15	COM	21/2/2011 11:57	COM
		Appropriate team composition and allocation		10		15	21/2/2011 13:23	COM	21/2/2011 16:38
			Full-time allocaton			2	2	18/2/2011 12:49	COM
			Mixed team			3	3	11/2/2011 14:58	COM
			Right people in the right place			1	1	18/2/2011 12:30	COM
			Small teams			4	5	18/2/2011 12:48	COM
			Whole team			4	4	11/2/2011 15:01	COM
		Conflicts in the team		2		2	18/2/2011 13:48	COM	16/3/2011 16:38
		Human relations investment		3		3	21/2/2011 11:51	COM	16/3/2011 16:37
			More fun and relaxing			2	2	16/3/2011 13:43	COM
		Shield the team from disturbances		3		4	17/2/2011 17:02	COM	16/3/2011 13:20
		Spend much time planning		1		1	18/2/2011 13:15	COM	18/2/2011 13:15
		Staff turnover		8		9	11/2/2011 15:05	COM	16/3/2011 14:03
		Strong customer participation		3		3	21/2/2011 13:43	COM	21/2/2011 17:43
			Customer participation			1	1	11/2/2011 13:39	COM
			Customers and developers as a single team			1	1	11/2/2011 15:46	COM
			Direct contact with the real customer			1	1	17/2/2011 17:50	COM
		Strong team commitment		3		3	18/2/2011 14:29	COM	16/3/2011 16:34
			Commitment with the product			1	1	16/3/2011 2:40	COM
		Task easiness		3		5	18/2/2011 17:46	COM	21/2/2011 12:50
			Known task			1	1	11/2/2011 13:19	COM
			Non-usual tasks			1	1	11/2/2011 14:47	COM
			Requirements similarity			1	1	11/2/2011 14:43	COM
			Task complexity			1	1	11/2/2011 15:32	COM
			Usual tasks			1	1	11/2/2011 14:53	COM
		Team cannot take some decisions		1		1	17/2/2011 20:17	COM	21/2/2011 12:34
		Team co-location		3		3	21/2/2011 13:41	COM	21/2/2011 17:43
			Co-location			3	3	11/2/2011 13:37	COM
		Team member playing just a specific role		1		1	18/2/2011 11:50	COM	21/2/2011 12:35
		Time to study		1		1	17/2/2011 17:49	COM	18/2/2011 17:52
			Lack of time to study			1	1	16/3/2011 13:16	COM

Negative factor	20	67	11/2/2011 13:07	COM	16/3/2011 16:38	COM	
Positive factor	22	112	11/2/2011 13:06	COM	16/3/2011 16:36	COM	
Rework	1	1	17/2/2011 17:24	COM	17/2/2011 17:24	COM	
Waste	1	1	11/2/2011 14:04	COM	11/2/2011 14:04	COM	

Appendix D

Action research protocol

This appendix describes the instruments we are using to collect data throughout the action research. The instrument often changes to incorporate new questions based on the research cycles.

D.1 Context Evaluation Instrument

Considering the importance of the context to understand productivity factors and monitoring, we are using a detailed protocol to describe companies context. The instrument is adapted from Kruchten [2011].

D.1.1 Organizational level: environmental factors

Business domain For what domain of activity is this organization developing software? Web-based systems, aerospace embedded systems, small hand-held instrumentation? Is software development the primary business activity of the organization, or at the other end, are we dealing with the IT organization of a business for which software is not at all the primary output. This factor, more than any of the other four, will condition, or constrain many of the project-level factors. For example, aerospace or biomedical instrumentation projects tend to be safety-critical.

Number of instances How many instances of the software system (large or small) will be actually deployed? Are you building one single system, a dozen, a thousand, or millions? One-off systems are often internal to an organization, or developed on demand by a system integrator.

Maturity of organization How long has that organization been developing software? How mature are the processes (and the people) relative to software development? Is the organization a small start-up, an SME (small or medium enterprise), or a large multinational firm? A small start-up is more likely to focus on a single, commercial piece of software, or more rarely open-source.

Level of innovation How innovative is the organization? Are you creators or early adopters of new ideas and technologies? Or, treading on very traditional grounds? Large IT organizations, or government agencies tend to be rather risk-adverse and rarely tread on unchartered territory.

Culture In which culture are the projects immersed? Are we speaking here of both national culture and corporate culture? What are the systems of values, beliefs, and behaviors that will impact, support, or interplay with the software development practices?

D.1.2 Project-level context attributes: the octopus model

Size The overall size of the system under development is, by far the greatest factor, as it will drive in turn the size of the team, the number of teams, the needs for communication and

coordination between teams, the impact of changes, and so on. Number of person-months, or size of the code, or development budget are all possible proxies for the size.

Stable architecture Is there an implicit, obvious, de facto architecture already in place at the start of the project? Most projects are not novel enough to require a lot of architectural effort. They follow commonly accepted patterns in their respective domain. Many of the key architectural decisions are done in the first few days, by choice of middleware, operating system, programming languages, and so on.

Business model What is the money flow? Are you developing an internal system, a commercial product, a bespoke system on contract for a customer, a component of a large system involving many different parties? Is it free, libre, and open-source software (FLOSS)?

Team distribution Linked often to the size of the project, how many teams are involved and are not collocated? This increases the need for more explicit communication and coordination of decisions, as well as more stable interfaces between teams and between the software components that they are responsible for.

Rate of change Although agile methods are "embracing changes", not all domains and system experience a very rapid pace of change in their environment. How stable is your business environment and how much risks (and unknowns) are you facing? There are still projects with very stable requirement definitions.

Age of system Are we looking at the evolution of a large legacy system, bringing in turn many hidden assumptions regarding the architecture, or the creation of a new system with fewer constraints?

Criticality How many people die or are hurt if the system fails? Documentation needs to increase dramatically to satisfy external agencies who will want to make sure that the safety of the public is assured.

Governance How are projects started, terminated? Who decides what happens when things go wrong? How is success or failure defined? Who manage the software project managers?

D.2 Monitoring and measurement usefulness protocol

Open-ended questions.

1. What do you consider agile team productivity?
2. How do you perceive the team productivity throughtout the project?
3. What are the main factors influencing the team productivity?
4. How do metrics are added or removed in your project?
5. Do you think that our project metrics generate overhead?
6. Do you think that our team metrics generate overhead?
7. Can you provide some specific example in your project in which metrics were added or removed? How did you analyze them?
8. How do metrics are helping your activities in the project?
9. How do metrics are helping team activities in the project?
10. Can you tell me some episode in the current project in which some metric helped you to: i) reflect on project results, discuss errors and/or unexpected results; ii) ask for feedback; iii) drive any type of change (either on yourself or on the team/project)?
11. Can you tell me some episode in the current project in which a metric had a negative outcome?
12. Would you use metrics in your next project? In which situations? Why?

13. Can you tell me some episode in the current project in which a metric had a unexpected outcome?

Close-ended questions - Scale ranged from 5 = "strongly agree" to 1 = "strongly disagree".

1. I could make a change in the project metrics if I really felt the need and if the change was justified
2. Learning to operate the metrics tool is easy for me
3. My role in the metrics program might affect my relationship with my co-workers
4. My schedule includes sufficient time for measurement-related activities
5. Team members have valid, well-defined reasons for introducing the metrics program
6. The metrics program activities are compatible with the existing work practices in the organization
7. I find the metrics tool flexible to interact with
8. It is easy for me to become skilled at using the metrics tool
9. Metrics data about my project will help the organization to learn something useful
10. My participation in such a metrics program will make my organization more successful
11. The metrics program involves more interaction with my colleagues and co-workers
12. The metrics tool is compatible and well integrated with the existing tools and applications
13. There are qualified, well-prepared people available to work on software metrics in my organization.
14. There is sufficient funding available for metrics-related activities in my organization
15. I find the metrics tool easy to use
16. I get to see the results that are relevant to me or my project
17. I have access to all the data I need for analysis of the collected metrics
18. I have to make some major changes in my way of working to adapt to the metrics tool
19. The data collection process is automated
20. There is a lot of discussion in the office, regarding the metrics program
21. I agree that the data is collected with the appropriate frequency
22. Metrics are incrementally added or removed througout the project
23. Metrics goals are congruent with business goals
24. My participation in the metrics program is voluntary
25. The proportion of effort on data collection is appropriate

D.3 Criteria for Action Research Evaluation

D.3.1 Criteria for the Principle of the Researcher–Client Agreement (RCA)

1. Did both the researcher and the client agree that CAR was the appropriate approach for the organizational situation?
2. Was the focus of the research project specified clearly and explicitly?
3. Did the client make an explicit commitment to the project?
4. Were the roles and responsibilities of the researcher and client organization members specified explicitly?
5. Were project objectives and evaluation measures specified explicitly?
6. Were the data collection and analysis methods specified explicitly?

D.3.2 Criteria for the Cyclical Process Model (CPM)

1. Did the project follow the CPM or justify any deviation from it?
2. Did the researcher conduct an independent diagnosis of the organizational situation?
3. Were the planned actions based explicitly on the results of the diagnosis?
4. Were the planned actions implemented and evaluated?
5. Did the researcher reflect on the outcomes of the intervention?
6. Was this reflection followed by an explicit decision on whether or not to proceed through an additional process cycle?
7. Were both the exit of the researcher and the conclusion of the project due to either the project objectives being met or some other clearly articulated justification?

D.3.3 Criteria for the Principle of Theory

1. Were the project activities guided by a theory or set of theories?
2. Was the domain of investigation, and the specific problem setting, relevant and significant to the interests of the researcher's community of peers as well as the client?
3. Was a theoretically based model used to derive the causes of the observed problem?
4. Did the planned intervention follow from this theoretically based model?
5. Was the guiding theory, or any other theory, used to evaluate the outcomes of the intervention?

D.3.4 Criteria for the Principle of Change through Action

1. Were both the researcher and client motivated to improve the situation?
2. Were the problem and its hypothesized cause(s) specified as a result of the diagnosis?
3. Were the planned actions designed to address the hypothesized cause(s)?
4. Did the client approve the planned actions before they were implemented?
5. Was the organization situation assessed comprehensively both before and after the intervention?
6. Were the timing and nature of the actions taken clearly and completely documented?

D.3.5 Criteria for the Principle of Learning through Reflection

1. Did the researcher provide progress reports to the client and organizational members?
2. Did both the researcher and the client reflect upon the outcomes of the project?
3. Were the research activities and outcomes reported clearly and completely?
4. Were the results considered in terms of implications for further action in this situation?
5. Were the results considered in terms of implications for action to be taken in related research domains?
6. Were the results considered in terms of implications for the research community (general knowledge, informing/re-informing theory)?
7. Were the results considered in terms of the general applicability of CAR?

D.4 Teamwork Effectiveness Evaluation Instrument

Team Design Measure

Instructions: This questionnaire consists of statements about your team and how your team functions as a group.
Please indicate the extent to which each statement describes your team by circling a number to the right of each statement.
** Leave blank if do not know or not applicable

Self-Management

	Strongly disagree	Disagree	Neither agree nor disagree	Agree	Strongly agree
The members of my team are responsible for determining the methods, procedures, and schedules with which the work gets done.	○	○	○	○	○
My team rather than my manager decides who does what tasks within the team.	○	○	○	○	○
Most work-related decisions are made by the members of my team rather than by my manager.	○	○	○	○	○

Participation

	Strongly disagree	Disagree	Neither agree nor disagree	Agree	Strongly agree
As a member of a team, I have a real say in how the	○	○	○	○	○

say in how the team carries out its work.					
Most members of my team get a chance to participate in decision making.					
My team is designed to let everyone participate in decision making.					

Task Variety

	Strongly disagree	Disagree	Neither agree nor disagree	Agree	Strongly agree
Most members of my team get a chance to learn the different tasks the team performs.					
Most everyone on my team gets a chance to do the more interesting tasks.					
Task assignments often change from day to day to meet the workload needs of the team.					

Task Significance (Importance)

	Strongly disagree	Disagree	Neither agree nor disagree	Agree	Strongly agree
The work performed by my team is important to the customers in my area.					
My team makes an important contribution to serving the company's customers.					
My team helps me feel that my work is important to the					

is important to the company.					

Task Identity (Mission)

	Strongly disagree	Disagree	Neither agree nor disagree	Agree	Strongly agree
The team concept allows all the work on a given product to be completed by the same set of people.					
My team is responsible for all aspects of a product for its area.					
My team is responsible for its own unique area or segment of the business.					

Task Interdependence

	Strongly disagree	Disagree	Neither agree nor disagree	Agree	Strongly agree
I cannot accomplish my tasks without information or materials from other members of my team.					
Other members of my team depend on me for information or materials needed to perform their tasks.					
Within my team, jobs performed by team members are related to one another.					

Goal Interdependence

	Strongly		Neither		Strongly

	disagree	Disagree	agree nor disagree	Agree	agree
My work goals come directly from the goals of my team.	○	○	○	○	○
My work activities on any given day are determined by my team's goals for that day.	○	○	○	○	○
I do very few activities on my job that are not related to the goals of my team.	○	○	○	○	○

Feedback and Rewards

	Strongly disagree	Disagree	Neither agree nor disagree	Agree	Strongly agree
Feedback about how well I am doing my job comes primarily from information about how well the entire team is doing.	○	○	○	○	○
My performance evaluation is strongly influenced by how well my team performs.	○	○	○	○	○
Many rewards from my job (pay, promotion, etc.) are determined in large part by my contributions as a team member.	○	○	○	○	○

Heterogeneity (Membership)

	Strongly disagree	Disagree	Neither agree nor disagree	Agree	Strongly agree
The members of my team vary widely in their areas of expertise	○	○	○	○	○

areas of expertise.

The members of my team have a variety of different backgrounds and experiences.	○	○	○	○	○
The members of my team have skills and abilities that complement each other.	○	○	○	○	○

Flexibility (Member Flexibility)

	Strongly disagree	Disagree	Neither agree nor disagree	Agree	Strongly agree
Most members of my team know each other's jobs.	○	○	○	○	○
It is easy for the members of my team to fill in for one another.	○	○	○	○	○
My team is very flexible in terms of membership.	○	○	○	○	○

Relative Size

	Strongly disagree	Disagree	Neither agree nor disagree	Agree	Strongly agree
The number of people in my team is too small for the work to be accomplished. (Reverse scored)	○	○	○	○	○

Preference for Team Work (Team Work Preferences)

	Strongly disagree	Disagree	Neither agree nor disagree	Agree	Strongly agree
If given the choice, I would prefer to work as part of a team rather than work alone.	○	○	○	○	○
I find that working as a member of a team increases my ability to perform	○	○	○	○	○

ability to perform effectively.	○	○	○	○	○
I generally prefer to work as part of a team.	○	○	○	○	○

Training

	Strongly disagree	Disagree	Neither agree nor disagree	Agree	Strongly agree
The company provides adequate technical training for my team.	○	○	○	○	○
The company provides adequate quality and customer service training for my team.	○	○	○	○	○
The company provides adequate team skills training for my team (communication, organization, interpersonal, etc.)	○	○	○	○	○

Managerial Support

	Strongly disagree	Disagree	Neither agree nor disagree	Agree	Strongly agree
Higher management in the company supports the concept of teams.	○	○	○	○	○
Mymanagersupportstheconceptofteams.	○	○	○	○	○

Communication/Cooperation between Work Groups

	Strongly disagree	Disagree	Neither agree nor disagree	Agree	Strongly agree
I frequently talk to other people in the company besides the people on my team.	○	○	○	○	○
There is little competition					

competition between my team and other teams in the company.	○	○	○	○	○
Teams in the company cooperate to get the work done.	○	○	○	○	○

Potency (Spirit)

	Strongly disagree	Disagree	Neither agree nor disagree	Agree	Strongly agree
Members of my team have great confidence that the team can perform effectively.	○	○	○	○	○
My team can take on nearly any task and complete it.	○	○	○	○	○
My team has a lot of team spirit.	○	○	○	○	○

Social Support

	Strongly disagree	Disagree	Neither agree nor disagree	Agree	Strongly agree
Being in my team gives me the opportunity to work in a team and provide support to other team members.	○	○	○	○	○
My team increases my opportunities for positive social interaction.	○	○	○	○	○
Members of my team help each other out at work when needed.	○	○	○	○	○

Workload Sharing (Sharing the Work)

	Strongly disagree	Disagree	Neither agree nor disagree	Agree	Strongly agree
Everyone on my team does their					

team does their fair share of the work.	○	○	○	○	○
No one in my team depends on other team members to do the work for them.	○	○	○	○	○
Nearly all the members of my team contribute equally to the work.	○	○	○	○	○

Communication/Cooperation within the Work Group

	Strongly disagree	Disagree	Neither agree nor disagree	Agree	Strongly agree
Members of my team are very willing to share information with other team members about our work.	○	○	○	○	○
Teams enhance the communications among people working on the same product.	○	○	○	○	○
Members of my team cooperate to get the work done.	○	○	○	○	○

Submit

Bibliography

Abbas *et al.***(2008)** Noura Abbas, Andrew M. Gravell and Gary B. Wills. Historical roots of agile methods: Where did "agile thinking" come from? In *Agile Processes in Software Engineering and Extreme Programming*, volume 9 of *Lecture Notes in Business Information Processing*, pages 94–103. Springer Berlin Heidelberg, 2008. ISBN 978-3-540-68255-4. Cited in page 12, 94

Abdel-Hamid(1989) T. K. Abdel-Hamid. The dynamics of software project staffing: A system dynamics based simulation approach. *IEEE Transactions on Software Engineering*, 15(2):109–119, 1989. Cited in page 36

Abdel-Hamid and Madnick(1991) Tarek Abdel-Hamid and Stuart E. Madnick. *Software project dynamics: an integrated approach*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1991. ISBN 0-13-822040-9. Cited in page 36, 89

Abernathy(1978) W. J. Abernathy. *The Productivity Dilemma: Roadblock to Innovation in the Automobile Industry*. Johns Hopkins University Press, 1978. Cited in page 18

Abrahamsson *et al.***(2009)** Pekka Abrahamsson, Kieran Conboy and Xiaofeng Wang. 'lots done, more to do': the current state of agile systems development research. *EJIS*, 18(4):281–284, 2009. Cited in page 17, 92, 93

Acuña *et al.***(2009)** Silvia T. Acuña, Marta Gómez and Natalia Juristo. How do personality, team processes and task characteristics relate to job satisfaction and software quality? *Inf. Softw. Technol.*, 51(3):627–639, March 2009. ISSN 0950-5849. doi: 10.1016/j.infsof.2008.08.006. URL http://dx.doi.org/10.1016/j.infsof.2008.08.006. Cited in page 71

Adler *et al.***(2009)** Paul S Adler, Mary Benner, David James, John Paul, Emi Osono, Bradley R Staats, Hirotaka Takeuchi, Michael L Tushman, Sidney G Winter, David James Brunner and et al. Perspectives on the productivity dilemma. *Journal of Operations Management*, 27(2): 99–113, 2009. Cited in page 17, 18, 19

Ambler(2008) Scott Ambler. Acceleration: An agile productivity measure. http://www.ibm.com/developerworks/mydeveloperworks/blogs/ambler/entry/metric_acceleration, Oct 2008. Cited in page 40, 41

Argyris and Schön(1978) Chris Argyris and Donald A Schön. Organizational learning: A theory of action perspective. *The Journal of Applied Behavioral Science*, 15(4):542–548, 1978. Cited in page 13

Arrow and McGrath(1995) H. Arrow and J. E. McGrath. Membership dynamics in groups at work: a theoretical framework. *Research in organizational behavior*, 17:373—-411, 1995. Cited in page 80

Atkins *et al.***(2000)** David L. Atkins, Audris Mockus and Harvey P. Siy. Measuring technology effects on software change cost. *Bell Labs Technical Journal*, pages 7–18, April–June 2000. Cited in page 32

Attride-Stirling(2001) Jennifer Attride-Stirling. Thematic networks: an analytic tool for qualitative research. *Qualitative Research*, 1(3):385–405, December 2001. Cited in page 77, 78

Augustine *et al.***(2005)** Sanjiv Augustine, Bob Payne, Fred Sencindiver and Susan Woodcock. Agile project management: steering from the edges. *Commun. ACM*, 48:85–89, December 2005. ISSN 0001-0782. Cited in page 72

Austin(1996) R.D. Austin. *Measuring and Managing Performance in Organizations*. Dorset House, New York, 1996. Cited in page 25, 43, 107, 115

Avison *et al.***(2001)** David E. Avison, Richard Baskerville and Michael D. Myers. Controlling action research projects. *IT & People*, 14(1):28–45, 2001. Cited in page 55, 123

Baburoglu and Ravn(1992) O N Baburoglu and I Ravn. Normative action research. *Organization Studies*, 13(1):19–34, 1992. URL http://oss.sagepub.com/content/13/1/019.short. Cited in page 55

Balijepally *et al.***(2009)** Venugopal Balijepally, RadhaKanta Mahapatra, Sridhar P. Nerur and Kenneth H. Price. Are two heads better than one for software development? the productivity paradox of pair programming. *MIS Quarterly*, 33(1):91–118, 2009. Cited in page 1, 85

Banker *et al.***(1991)** R. D. Banker, S. M. Datar and C. F. Kemerer. A model to evaluate variables impacting the productivity of software maintenance projects. *Management Science*, 23:1–18, 1991. Cited in page 33, 37

Barlow and Möller(1996) J. Barlow and C. Möller. *Complaint Is a Gift: Using Customer Feedback As a Strategic Tool*. Berrett-koehler Series. Berrett-Koehler, 1996. Cited in page 108

Basili(1990) Victor R. Basili. Viewing maintenance as reuse-oriented software development. *IEEE Software*, 7(1):19–25, 1990. Cited in page 95

Basili and Weiss(1984) Victor R. Basili and David M. Weiss. A methodology for collecting valid software engineering data. *IEEE Trans. Software Eng.*, 10(6):728–738, 1984. Cited in page 118

Basili *et al.***(1996)** Victor R Basili, Lionel C. Briand and Walcélio L. Melo. How reuse influences productivity in object-oriented systems. *Communications of the ACM*, 39(10):104–116, 1996. Cited in page 32

Baskerville and Wood-Harper(1998) R. Baskerville and A. T. Wood-Harper. Diversity in information systems action research methods. *European Journal of Information Systems*, 7:90–107, June 1998. Cited in page 55

Baskerville and Myers(2004) Richard Baskerville and Michael D Myers. Special issue on action research in information systems: Making is research relevant to practice - foreword. *MIS Quarterly*, 28(3):329–335, 2004. Cited in page 55

Baskerville *et al.***(2006)** Richard Baskerville, Balasubramaniam Ramesh, Linda Levine and Jan Pries-Heje. High-speed software development practices: What works, what doesn't. *IT Professional*, 8(4):29–36, 2006. Cited in page 11, 21

Baskerville(1999) Richard L. Baskerville. Investigating information systems with action research. *Communications of the AIS*, 2(3es), November 1999. Cited in page 98

Baskerville and Wood-Harper(1996) Richard L Baskerville and A Trevor Wood-Harper. A critical perspective on action research as a method for information systems research. *Journal of Information Technology*, 11(3):235–246, 1996. URL http://www.palgrave-journals.com/doifinder/10.1080/026839696345289. Cited in page 99, 101

Beck(1999) Kent Beck. *Extreme Programming Explained: Embrace Change*. Addison-Wesley Professional, 1st edition, October 1999. ISBN 0201616416. Cited in page 65

Beck and Andres(2004) Kent Beck and Cynthia Andres. *Extreme Programming Explained : Embrace Change (2nd Edition)*. Addison-Wesley Professional, November 2004. ISBN 0321278658. Cited in page 1, 31, 40, 41, 72, 74, 87, 88, 105

Beck and Boehm(2003) Kent Beck and Barry W. Boehm. Agility through discipline: A debate. *IEEE Computer*, 36(6):44–46, 2003. Cited in page 17

Beck *et al.***(2001)** Kent Beck, Mike Beedle, Arie van Bennekum, Alistair Cockburn, Ward Cunningham, Martin Fowler, James Grenning, Jim Highsmith, Andrew Hunt, Ron Jeffries, Jon Kern, Brian Marick, Robert C. Martin, Steve Mellor, Ken Schwaber, Jeff Sutherland and Dave Thomas. Manifesto for agile software development. http://agilemanifesto.org/, 2001. Cited in page 8, 12, 13, 38, 71, 79

Beecham *et al.***(2008)** Sarah Beecham, Nathan Baddoo, Tracy Hall, Hugh Robinson and Helen Sharp. Motivation in software engineering: A systematic literature review. *Information & Software Technology*, 50(9-10):860–878, 2008. Cited in page 34, 35, 79, 93

Begel and Nagappan(2007) Andrew Begel and Nachiappan Nagappan. Usage and perceptions of agile software development in an industrial context: An exploratory study. In *Proceedings of the First International Symposium on Empirical Software Engineering and Measurement*, pages 255–264, Washington, DC, USA, 2007. IEEE Computer Society. Cited in page 68

Begel *et al.***(2009)** Andrew Begel, Nachiappan Nagappan, Christopher Poile and Lucas Layman. Coordination in large-scale software teams. In *Proceedings of the 2009 ICSE Workshop on Cooperative and Human Aspects on Software Engineering*, CHASE '09, pages 1–7, Washington, DC, USA, 2009. IEEE Computer Society. ISBN 978-1-4244-3712-2. Cited in page 92

Behfar *et al.***(2011)** Kristin J. Behfar, Elizabeth A. Mannix, Randall S. Peterson and William M. Trochim. Conflict in small groups: The meaning and consequences of process conflict. *Small Group Research*, 42(2):127–176, 2011. Cited in page 91, 94

Behrens(2009) P. Behrens. Measuring agility: Top 5 metrics and myths. http://pm.versionone.com/Webinar_MetricsMyths.html, 2009. VersionOne. Cited in page 40, 41

Bell(2007) Suzanne T. Bell. Deep-level composition variables as predictors of team performance: A meta-analysis. *Journal of Applied Psychology*, 92(3):595–615, 2007. Cited in page 72

Bititci *et al.***(2011)** Umit Bititci, Patrizia Garengo, Viktor Dörfler and Sai Nudurupati. Performance measurement: Challenges for tomorrow. *International Journal of Management Reviews*, page in press, 2011. ISSN 1468-2370. doi: 10.1111/j.1468-2370.2011.00318.x. URL http://dx.doi.org/10.1111/j.1468-2370.2011.00318.x. Cited in page 24, 26, 30, 31, 121

Blackburn *et al.***(2000)** Joseph Blackburn, Gary Scudder and Luk N. Van Wassenhove. Concurrent software development. *Communications of ACM*, 43:200–214, November 2000. ISSN 0001-0782. Cited in page 33, 34, 35, 36, 37, 72, 91

Blackburn *et al.***(1996)** Joseph D. Blackburn, Gary D. Scudder and Luk N. Van Wassenhove. Improving speed and productivity of software development: A global survey of software developers. *IEEE Transactions on Software Engineering*, 22:875–885, December 1996. ISSN 0098-5589. Cited in page 34, 36, 37, 91

Boehm(1981) Barry Boehm. *Software Engineering Economics*. Prentice Hall, 1981. Cited in page 34, 35, 36, 37, 72

Boehm and Turner(2005) Barry Boehm and Richard Turner. *Balancing Agility and Discipline: A Guide for the Perplexed*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2005. ISBN 0321186125. Cited in page 11

Boehm(1987) Barry W. Boehm. Improving software productivity. *Computer*, 20(9):43–57, 1987. ISSN 0018-9162. doi: http://dx.doi.org/10.1109/MC.1987.1663694. Cited in page 1, 36

Boehm(2000) Barry W. Boehm. *Software Cost Estimation with Cocomo II*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 1st edition, 2000. ISBN 0130266922. Cited in page 32, 33, 34, 36, 37

Bogsnes(2008) B. Bogsnes. *Implementing Beyond Budgeting: Unlocking the Performance Potential*. John Wiley & Sons, 2008. ISBN 9780470466193. Cited in page 43

Bonner *et al.***(2010)** Nancy A. Bonner, James T. C. Teng and Sridhar P. Nerur. The perceived advantage of agile development methodologies by software professionals: Testing an innovation-theoretic model. In *AMCIS*, page 93, 2010. Cited in page 15

Bosch-Sijtsema *et al.***(2009)** Petra Bosch-Sijtsema, Virpi Ruohomäki and Matti Vartiainen. Knowledge work productivity in distributed teams. *Journal of Knowledge Management*, 13(6): 533–546, 2009. Cited in page 22, 38, 76

Bourne *et al.***(2000)** Mike Bourne, John Mills, Mark Wilcox, Andy Neely and Ken Platts. Designing, implementing and updating performance measurement systems. *International Journal of Operations & Production Management*, 20(7):754–771, 2000. Cited in page xi, 30, 31, 121

Boyatzis(1998) R. E. Boyatzis. *Transforming qualitative information: thematic analysis and code development*. Sage Publications, 1998. Cited in page 76

Braun and Clarke(2006) V. Braun and V. Clarke. Using thematic analysis in psychology. *Qualitative Research in Psychology*, 3(2):77–101, 2006. Cited in page 76, 77

Brian R Dineen(2003) Raymond A Noe Brian R Dineen. The impact of team fluidity and its implications for human resource management research and practice. *Research in Personnel and Human Resources Management*, 22:1–37, 2003. Cited in page 90

Briand *et al.***(1996)** L C Briand, C M Differding and H D Rombach. Practical guidelines for measurement-based process improvement. *Software Process: Improvement and Practice*, 2(4): 253–280, 1996. Cited in page 23

Brooks(1975) F. P. Brooks. *The Mythical Man-Month: Essays on Software Engineering*. Addison-Wesley, 1975. Cited in page 12

Brooks(1995) Frederick P. Brooks, Jr. *The mythical man-month (anniversary ed.)*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1995. ISBN 0-201-83595-9. Cited in page 36, 91

Brown *et al.***(2010)** Nanette Brown, Yuanfang Cai, Yuepu Guo, Rick Kazman, Miryung Kim, Philippe Kruchten, Erin Lim, Alan MacCormack, Robert Nord, Ipek Ozkaya, Raghvinder Sangwan, Carolyn Seaman, Kevin Sullivan and Nico Zazworka. Managing technical debt in software-reliant systems. In *Proceedings of the FSE/SDP workshop on Future of software engineering research*, FoSER'10, pages 47–52, 2010. Cited in page 110

Bruckhaus *et al.***(1996)** Tilmann Bruckhaus, Nazim H. Madhavji, Ingrid Janssen and John Henshaw. The impact of tools on software productivity. *IEEE Softw.*, 13(5):29–38, September 1996. ISSN 0740-7459. doi: 10.1109/52.536456. URL http://dx.doi.org/10.1109/52.536456. Cited in page 37

Brudan(2010) Aurel Brudan. Rediscovering performance management: systems, learning and integration. *Measuring Business Excellence*, 14(1):109–123, 2010. Cited in page xiii, 26, 121

Campion *et al.***(1993)** Michael A Campion, Gina J Medsker and A Catherine Higgs. Relations between work group characteristics and effectiveness: Implications for designing effective work groups. *Personnel Psychology*, 46(4):823–847, 1993. Cited in page xii, 100, 112, 114, 115, 116, 117

Carey and Kacmar(1997) J.M. Carey and C.J. Kacmar. The impact of communication mode and task complexity on small group performance and member satisfaction. *Comput. Hum. Behav.*, 13(1):23–49, 1997. Cited in page 36

Cascio(1991) F. W. Cascio. *Managing Human Resource: Productivity, Quality of Work Life and Profits*. New York: McGraw Hill, 3rd edition edition, 1991. Cited in page 80

Castka *et al.*(2001) P Castka, C J Bamber, J M Sharp and P Belohoubek. Factors affecting successful implementation of high performance teams. *Team Performance Management*, 7(7/8): 123–134, 2001. Cited in page 38, 39

Checkland(1991) P Checkland. *From framework through experience to learning: the essential nature of action research*, pages 397–403. North-Holland, 1991. Cited in page 55

Chow and Cao(2008) T. Chow and D.-B. Cao. A survey study of critical success factors in agile software projects. *Journal of Systems and Software*, 81(6):961–971, 2008. Cited in page 72, 91

Cockburn(2006) A. Cockburn. *Agile software development: the cooperative game (agile software development series)*. Addison-Wesley Professional, 2006. Cited in page 15, 40

Cockburn and Highsmith(2001) Alistair Cockburn and Jim Highsmith. Agile software development: The people factor. *Computer*, 34:131–133, 2001. ISSN 0018-9162. Cited in page 13, 94

Cohen(1988) J Cohen. *Statistical Power Analysis for the Behavioral Sciences (2nd Edition)*. Routledge Academic, 1988. URL http://www.amazon.com/Statistical-Power-Analysis-Behavioral-Sciences/dp/0805802835. Cited in page 62

Cohen and Bailey(1997) Susan G. Cohen and Diane E. Bailey. What makes teams work: Group effectiveness research from the shop floor to the executive suite. *Journal of Management*, 23(3): 239–290, 1997. Cited in page 8, 71, 72, 91

Cohen and Ledford(1994) Susan G. Cohen and Gerald E. Ledford. The effectiveness of self-managing teams: A quasi-experiment. *Human Relations*, 47(1):13–43, 1994. Cited in page 72

Cohn(2005) Mike Cohn. *Agile Estimating and Planning*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 2005. ISBN 0131479415. Cited in page 40, 41

Conboy(2009) Kieran Conboy. Agility from first principles: Reconstructing the concept of agility in information systems development. *Information Systems Research*, 20(3):329–354, 2009. doi: 10.1287/isre.1090.0236. URL http://isr.journal.informs.org/content/20/3/329.abstract. Cited in page 15, 16, 17

Conboy and Fitzgerald(2004) Kieran Conboy and Brian Fitzgerald. Toward a conceptual framework of agile methods: a study of agility in different disciplines. In *Proceedings of the 2004 ACM workshop on Interdisciplinary software engineering research*, WISER '04, pages 37–44, New York, NY, USA, 2004. ACM. ISBN 1-58113-988-8. Cited in page 15, 16

Conboy *et al.*(2011) Kieran Conboy, Sharon Coyle, Xiaofeng Wang and Minna Pikkarainen. People over process: Key challenges in agile development. *IEEE Software*, 28(4):48–57, 2011. Cited in page 79

Coram and Bohner(2005) Michael Coram and Shawn Bohner. The impact of agile methods on software project management. In *Proceedings of the 12th IEEE International Conference and Workshops on Engineering of Computer-Based Systems*, pages 363–370, Washington, DC, USA, 2005. IEEE Computer Society. ISBN 0-7695-2308-0. Cited in page 89

Corbin and Strauss(2007) Juliet Corbin and Anselm C. Strauss. *Basics of Qualitative Research: Techniques and Procedures for Developing Grounded Theory*. Sage Publications, Inc, 3rd edition, 2007. Cited in page 77, 79

Corbin and Strauss(1990) Juliet M. Corbin and Anselm Strauss. Grounded theory research: Procedures, canons, and evaluative criteria. *Qualitative Sociology*, 13:3–21, 1990. ISSN 0162-0436. Cited in page 78

Corbucci *et al.***(2011)** Hugo Corbucci, Alfredo Goldman, Eduardo Katayama, Fabio Kon, Claudia Melo and Viviane Santos. Genesis and evolution of the agile movement in brazil – perspective from academia and industry. In *Proceedings of the 25th Brazilian Symposium on Software Engineering*, SBES'11, pages 98–107, Washington, DC, USA, 2011. IEEE Computer Society. Cited in page 12

Creswell(2003) John W. Creswell. *Research design : qualitative, quantitative, and mixed method approaches*. Sage Publications, 2nd ed edition, 2003. Cited in page 47

Cruzes and Dybå(2011) Daniela S. Cruzes and Tore Dybå. Recommended steps for thematic synthesis in software engineering (forthcoming on ieee-cs press). In *Proc. 5th IEEE conference on Empirical Software Engineering and Measurement (ESEM'2011)*, Banff, Canada, 2011. Cited in page 95

Cruzes and Dybå(2011) Daniela S. Cruzes and Tore Dybå. Research synthesis in software engineering: A tertiary study. *Information and Software Technology*, 53(5):440 – 455, 2011. Cited in page 76

Cunningham(1997) J. Barton Cunningham. Case study principles for different types of cases. *Quality & Quantity*, 31:401–423, 1997. ISSN 0033-5177. Cited in page xiii, 52

Cunningham(1992) Ward Cunningham. The wycash portfolio management system. In *Addendum to the proceedings on Object-oriented programming systems, languages, and applications*, OOPSLA '92, pages 29–30, New York, NY, USA, 1992. ACM. Cited in page 110

Dale and van der Zee(1992) CJ Dale and H van der Zee. Software productivity metrics: who needs them? *Information and Software Technology*, 34(11):731 – 738, 1992. Cited in page 21

Damm *et al.***(2006)** Lars-Ola Damm, Lars Lundberg and Claes Wohlin. Faults-slip-through - a concept for measuring the efficiency of the test process. *Software Process: Improvement and Practice*, 11(1):47–59, 2006. Cited in page 40, 41

Davenport and Harris(2007) T.H. Davenport and J.G. Harris. *Competing on analytics: the new science of winning*. Harvard Business Review. Harvard Business School Press, 2007. ISBN 9781422103326. URL http://books.google.com.br/books?id=n7Gp7Q84hcsC. Cited in page 26, 121

Davison *et al.***(2004)** Robert Davison, Maris G Martinsons and Ned Kock. Principles of canonical action research. *Information Systems Journal*, 14(1):65–86, 2004. Cited in page 54, 55, 57, 101

de Haas and Kleingeld(1999) Marco de Haas and Ad Kleingeld. Multilevel design of performance measurement systems: enhancing strategic dialogue throughout the organization. *Management Accounting Research*, 10(3):233 – 261, 1999. ISSN 1044-5005. Cited in page 29, 30

DeMarco and Lister(1999) T. DeMarco and T.R. Lister. *Peopleware: productive projects and teams*. Dorset House Pub., 1999. ISBN 9780932633439. Cited in page 78

DeMarco(1986) Tom DeMarco. *Controlling Software Projects: Management, Measurement, and Estimates*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 1986. ISBN 0131717111. Cited in page 26

DeMarco(2009) Tom DeMarco. Software engineering: An idea whose time has come and gone? *IEEE Software*, 26(4):96–95, 2009. Cited in page 26

Deming(1986) William Edwards Deming. *Out of the crisis*. MIT Press, 1986. Cited in page 26

Dickinson and McIntyre(1997) Terry L Dickinson and Robert M McIntyre. *A conceptual framework for teamwork measurement*, pages 19–43. Lawrence Earlbaum Associates, 1997. Cited in page 39

Dillman(2006) D. A. Dillman. *Mail and Internet Surveys: The Tailored Design Method 2007 Update with New Internet, Visual, and Mixed-Mode Guide*. John Wiley & Sons, 2006. ISBN 9780470080078. Cited in page 48, 49, 50, 51

Dingsøyr *et al.***(2010)** Torgeir Dingsøyr, Tore Dybå and Nils Brede Moe. *Agile Software Development: Current Research and Future Directions*. Springer Publishing Company, Incorporated, 1st edition, 2010. ISBN 3642125743, 9783642125744. Cited in page 11

Dorairaj *et al.***(2012)** Siva Dorairaj, James Noble and Petra Malik. Distribution and agility: It's all about trust. Technical Report 01, Victoria University of Wellington, January 2012. Cited in page 108

Dreu and Weingart(2003) Carsten K. W. De Dreu and Laurie R. Weingart. Task versus relationship conflict, team performance, and team member satisfaction: A meta-analysis. *Journal of Applied Psychology*, 88(4):741 – 749, 2003. Cited in page 91

Drucker(1994) P. F. Drucker. *Adventures of a Bystander*. Transaction Publishers, New Brunswick, NJ, 1994. Cited in page 22, 23, 121

Drucker(1999) P. F. Drucker. Knowledge-Worker Productivity: The Biggest Challenge. *California Management Review*, 41(2):79–94, 1999. Cited in page 22, 23, 29

Dybå *et al.***(2007)** Tore Dybå, Erik Arisholm, Dag I. K. Sjøberg, Jo E. Hannay and Forrest Shull. Are two heads better than one? on the effectiveness of pair programming. *IEEE Software*, 24: 12–15, November 2007. ISSN 0740-7459. Cited in page 1, 84

Dybå and Dingsoyr(2008) T. Dybå and T. Dingsoyr. Empirical studies of agile software development: A systematic review. *Information and Software Technology*, 50(9-10):833–859, 2008. ISSN 0950-5849. doi: http://dx.doi.org/10.1016/j.infsof.2008.01.006. Cited in page 2, 11, 42

Easterbrook *et al.***(2008)** Steve Easterbrook, Janice Singer, Margaret-anne Storey and Daniela Damian. Selecting empirical methods for software engineering research. In Forrest Shull, Janice Singer and Dag Sjøberg, editors, *Guide to Advanced Empirical Software Engineering*, pages 285–311. Springer, 2008. Cited in page 52

Eccles *et al.***(2010)** M. Eccles, J. Smith, M. Tanner, J. Van Belle and S. van der Watt. The impact of collocation on the effectiveness of agile is development teams. *Communications of the IBIMA*, 1:1–10, 2010. Cited in page 86, 92

Erickson *et al.***(2005)** John Erickson, Kalle Lyytinen and Keng Siau. Agile modeling, agile software development, and extreme programming: The state of research. *Journal of Database Management*, 16(4):88–100, 2005. URL http://www.igi-pub.com/articles/details.asp?ID=5327. Cited in page 15, 16

Faraj and Sproull(2000) Samer Faraj and Lee Sproull. Coordinating expertise in software development teams. *Management Science*, 46(12):pp. 1554–1568, 2000. Cited in page 34, 36, 37, 71

Fenton and Pfleeger(1997) Norman E. Fenton and Shari Lawrence Pfleeger. *Software Metrics: A Rigorous and Practical Approach*. PWS Publishing Co., Boston, MA, USA, 1997. ISBN 0534954251. Cited in page 23, 28, 29, 32, 33

Fereday and Muir-Cochrane(2006) Jennifer Fereday and E Muir-Cochrane. Demonstrating rigor using thematic analysis : A hybrid approach of inductive and deductive coding and theme development. *International Journal of Qualitative Methods*, 5(1):1–11, 2006. Cited in page 76

Fine and Porteus(1987) Charles H. Fine and Evan L. Porteus. Dynamic process improvement. Working papers 1952-87., Massachusetts Institute of Technology (MIT), Sloan School of Management, 1987. URL http://ideas.repec.org/p/mit/sloanp/2189.html. Cited in page 18

Fink(2003) Arlene Fink. *The Survey Handbook*. Sage Books, 2 edition, 2003. Volume 1 of the Survey Kit. Cited in page 48, 49, 50

Flin and Yule(2004) R Flin and S Yule. Leadership for safety: industrial experience. *Quality safety in health care*, 13(Suppl 2):ii45–ii51, 2004. Cited in page 72

Fowler(2003) Martin Fowler. Cannot measure productivity. http://martinfowler.com/bliki/CannotMeasureProductivity.html, August 2003. Cited in page 28, 29

Frakes and Succi(2001) W.B. Frakes and G. Succi. An industrial study of reuse, quality, and productivity. *Journal of Systems and Software*, 57:99–106, 2001. Cited in page 32, 33

Galbraith(1973) Jay R. Galbraith. *Designing Complex Organizations*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1st edition, 1973. ISBN 0201025590. Cited in page 3, 74, 87, 93, 101

Gat(2010) Israel Gat. *The Concise Executive Guide to Agile*. IEEE Computer Society Press, 2010. Cited in page 40, 41

George and Williams(2004) Boby George and Laurie Williams. A structured experiment of test-driven development. *Information and Software Technology*, 46(5):337–342, 2004. Cited in page 17

Gerring(2006) John Gerring. *Case Study Research: Principles and Practices*. Cambridge University Press, 2006. Cited in page 51, 52, 73

Goldman *et al.*(1991) S. Goldman, R.N. Nagel, K. Preiss, R. Dove, Iacocca Institute of Lehigh University and United States. Dept. of Defense. Office of Managing Technology. *21st Century Manufacturing Enterprise Strategy: Infrastructure*. 21st Century Manufacturing Enterprise Strategy. Iacocca Institute, Lehigh University, 1991. ISBN 9780962486630. Cited in page 14

Groves(1989) Robert M. Groves. *Survey Errors and Survey Costs*. John Wiley & Sons, New York, 1989. Cited in page 48, 51

Guinan *et al.*(1998) P. J. Guinan, J. Cooprider and S. Faraj. Enabling software development team performance during requirements definition: A behavioral versus technical approach. *Inform. Systems Research*, 9:101–125, 1998. Cited in page 37

Guzzo and Dickson(1996) Richard A. Guzzo and Marcus W. Dickson. Teams in Organizations: Recent Research on Performance and Effectiveness. *Annual Review of Psychology*, 47(1):307–338, 1996. Cited in page 71

Hall *et al.*(2008) Tracy Hall, Helen Sharp, Sarah Beecham, Nathan Baddoo and Hugh Robinson. What do we know about developer motivation? *IEEE Software*, 25(4):92–94, 2008. Cited in page 86

Hamilton *et al.***(2003)** Barton H Hamilton, Jack A Nickerson and Hideo Owan. Team incentives and worker heterogeneity : An empirical analysis of the impact of teams on productivity and participation. *Journal of Political Economy*, 111(3):465–497, 2003. Cited in page 92

Hannay and Benestad(2010) Jo E. Hannay and Hans Christian Benestad. Perceived productivity threats in large agile development projects. In *Proceedings of the 2010 ACM-IEEE International Symposium on Empirical Software Engineering and Measurement*, ESEM '10, pages 1–10, New York, NY, USA, 2010. ACM. Cited in page 92

Hartmann and Dymond(2006) Deborah Hartmann and Robin Dymond. Appropriate agile measurement: Using metrics and diagnostics to deliver business value. In *AGILE Conference*, pages 126–134, 2006. ISBN 0-7695-2562-8. Cited in page 29

Helo(2004) Petri Helo. Managing agility and productivity in the electronics industry. *Industrial Management Data Systems*, 104(7):567–577, 2004. Cited in page xi, 17, 18, 19

Henderson-Sellers and Serour(2005) Brian Henderson-Sellers and M. K. Serour. Creating a dual-agility method: The value of method engineering. *Journal of Database Management*, 16(4): 1–24, 2005. Cited in page 15

Herzberg(1993) Frederick Herzberg. *Motivation To Work*. Transaction Publishers, 1993. Cited in page 35

Highsmith(2007) J. Highsmith. Agile transitions, part 1. http://www.cutter.com/project/fulltext/advisor/2007/apm071108.html, 2007. Cited in page 11

Highsmith(1997) Jim Highsmith. Messy, exciting, and anxiety-ridden: Adaptive software development. American Programmer, v. 10, Jan 1997. Cited in page 17

Highsmith(2002) Jim Highsmith. What is agile software development? *CrossTalk - The Journal of Defense Software Engineering*, 1:4–9, October 2002. Cited in page 12

Highsmith(2004) Jim Highsmith. *Agile Project Management - Creating Innovative Products*. Pearson Education, Boston, 2004. Cited in page 15, 72

Highsmith(2006) Jim Highsmith. An adaptive performance management system. www.infoq.com/resource/articles/Adaptive-Performance-Management/en/resources/apms0606.pdf, August 2006. Cited in page 2, 42, 43

Highsmith and Cockburn(2001) Jim Highsmith and Alistair Cockburn. Agile software development: The business of innovation. *IEEE Computer*, 34(9):120–122, 2001. Cited in page 13

Hinds and Kiesler(2002) Pamela J. Hinds and Sara Kiesler, editors. *Distributed Work*. MIT Press, Cambridge, MA, USA, 2002. ISBN 0262083051. Cited in page 86, 92

Hoda *et al.***(2010)** Rashina Hoda, James Noble and Stuart Marshall. Organizing self-organizing teams. In *Proceedings of the 32nd ACM IEEE International Conference on Software Engineering - Volume 1*, ICSE '10, pages 285–294, New York, NY, USA, 2010. ACM. Cited in page 90, 91

Hope and Fraser(2003) J. Hope and R. Fraser. *Beyond budgeting: how managers can break free from the annual performance trap*. Harvard Business School Press, 2003. ISBN 9781578518661. Cited in page 43

Iivari and Iivari(2011) Juhani Iivari and Netta Iivari. The relationship between organizational culture and the deployment of agile methods. *Information and Software Technology*, 53(5):509 – 520, 2011. Cited in page 17

Ikonen *et al.***(2010)** Marko Ikonen, Petri Kettunen, Nilay Oza and Pekka Abrahamsson. Exploring the sources of waste in kanban software development projects. In *Proceedings of the 2010 36th EUROMICRO Conference on Software Engineering and Advanced Applications*, SEAA'10, pages 376–381, Washington, DC, USA, 2010. IEEE Computer Society. Cited in page 41, 104

Iversen *et al.***(2004)** Jakob H Iversen, Lars Mathiassen and Peter Axel Nielsen. Managing risk in software process improvement: An action research approach. *MIS Quarterly*, 28(3):395–433, 2004. Cited in page xiii, 55, 56, 99, 123

Jiang and Comstock(2007) Z. Jiang and C. Comstock. The factors significant to software development productivity. *World Academy of Science, Engineering and Technology,*, 25:pp. 160–164, Jan. 2007. Cited in page 37

Jiang *et al.***(2007)** Z. Jiang, P. Naudé and C. Comstock. An investigation on the variation of software development productivity. *International Journal of Computer, Information, and Systems Sciences, and Engineering*, 1(2):72–81, 2007. Cited in page 37

Jones(2008) Caper Jones. *Applied software measurement: Global Analysis of Productivity and Quality*. McGraw-Hill Osborne Media, 3rd edition, April 2008. Cited in page 24, 28

Jones(2000) Capers Jones. *Software assessments, benchmarks, and best practices*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2000. Cited in page 37

Joshi *et al.***(2009)** Aparna Joshi, Niti Pandey and Guohong (Helen) Han. Bracketing team boundary spanning: An examination of task-based, team-level, and contextual antecedents. *Journal of Organizational Behavior*, 30(6):731–759, 2009. Cited in page 92

Junior and Meira(2009) Gibeon Soares de Aquino Junior and Silvio Romero de Lemos Meira. Towards effective productivity measurement in software projects. In *Proceedings of the 2009 Fourth International Conference on Software Engineering Advances*, ICSEA'09, pages 241–249, Washington, DC, USA, 2009. IEEE Computer Society. Cited in page 40

Kacmar *et al.***(2006)** K M Kacmar, M C Andrews, D L Van Rooy, R C Steilberg and S Cerrone. Sure everyone can be replaced ... but at what cost? turnover as a predictor of unit-level performance. *Academy of Management Journal*, 49(1):133–144, 2006. Cited in page 90

Kadary(1992) V. Kadary. On application of earned value index to software productivity metrics in embedded computer systems. In *Proceedings of the Conference on Computer Systems and Software Engineering (CompEuro)*, page 666–670, 1992. Cited in page 28

Karlström and Runeson(2006) Daniel Karlström and Per Runeson. Integrating agile software development into stage-gate managed product development. *Empirical Software Engineering*, 11: 203–225, June 2006. ISSN 1382-3256. Cited in page 1

Katzenbach and Smith(2005) Jon R. Katzenbach and Douglas K. Smith. *The Wisdom of Teams: Creating the High-Performance Organization*. McGraw-Hill Professional, european version edition, September 2005. ISBN 0875843670. Cited in page 38

Kelley *et al.***(2003)** Kate Kelley, Belinda Clark, Vivienne Brown and John Sitzia. Good practice in the conduct and reporting of survey research. *International journal for quality in health care journal of the International Society for Quality in Health Care ISQua*, 15(3):261–266, 2003. URL http://www.ncbi.nlm.nih.gov/pubmed/12803354. Cited in page 48, 50

Kelly(2008) Allan Kelly. *Changing Software Development: Learning to Become Agile*. Wiley Publishing, 2008. ISBN 047051504X, 9780470515044. Cited in page 72

Kettunen(2009) Petri Kettunen. *Agile software development in large-scale new product development organization: team-level perspective*. PhD thesis, Helsinki University of Technology, 2009. Cited in page 11, 15, 55

Kitchenham *et al.*(2002) B.A. Kitchenham, S.L. Pfleeger, L.M. Pickard, P.W. Jones, D.C. Hoaglin, K. El Emam and J. Rosenberg. Preliminary guidelines for empirical research in software engineering. *Software Engineering, IEEE Transactions on*, 28(8):721 – 734, aug 2002. ISSN 0098-5589. doi: 10.1109/TSE.2002.1027796. Cited in page 74

Kitchenham(2004) Barbara Kitchenham. Software productivity measurement using multiple size measures. *IEEE Transactions on Software Engineering*, 30(12):1023–1035, 2004. ISSN 0098-5589. doi: http://dx.doi.org/10.1109/TSE.2004.104. Member-Mendes, Emilia. Cited in page 47

Kitchenham(2010) Barbara Kitchenham. What's up with software metrics? - a preliminary mapping study. *Journal of Systems and Software*, 83(1):37–51, January 2010. ISSN 0164-1212. Cited in page 21

Kitchenham *et al.*(2007) Barbara Kitchenham, David Ross Jeffery and Colin Connaughton. Misleading metrics and unsound analyses. *IEEE Software*, 24(2):73–78, 2007. Cited in page 4, 25, 28

Kitchenham and Pfleeger(2008) Barbara A. Kitchenham and Shari L. Pfleeger. Personal opinion surveys. In Forrest Shull, Janice Singer and Dag I. K. Sj/oberg, editors, *Guide to Advanced Empirical Software Engineering*, pages 63–92. Springer London, 2008. ISBN 978-1-84800-044-5. Cited in page 48, 49, 50, 61

Klein and Myers(1999) Heinz K. Klein and Michael D. Myers. A set of principles for conducting and evaluating interpretive field studies in information systems. *MIS Q.*, 23:67–93, March 1999. ISSN 0276-7783. Cited in page 4

Klepper(1996) Steven Klepper. Entry, exit, growth, and innovation over the product life cycle. *American Economic Review*, 86(3):562–583, 1996. URL http://www.jstor.org/stable/2118212. Cited in page 100

Kock(2004) Ned Kock. The three threats of action research: a discussion of methodological antidotes in the context of an information systems study. *Decision Support Systems*, 37(2):265–286, May 2004. ISSN 0167-9236. Cited in page 57

Kozlowski *et al.*(2008) S. Kozlowski, D. Watola, J Nowakowski, B. Kim and I. Botero. *Team effectiveness in complex organizations: cross-disciplinary perspectives and approaches*, chapter Developing adaptive teams: A theory of dynamic team leadership. SIOP frontiers series. Routledge, Mahwah, NJ, 2008. Cited in page 38

Kozlowski and Ilgen(2006) Steve W.J. Kozlowski and Daniel R. Ilgen. Enhancing the effectiveness of work groups and teams. *Psychological Science in the Public Interest*, 7(3):77–124, 2006. Cited in page 96

Krishnan(1998) M.S. Krishnan. The role of team factors in software cost and quality: An empirical analysis. *Inform. Technol People*, 1:20–35, 1998. Cited in page 32, 33, 34

Kruchten(2010) Philippe Kruchten. *Software architecture and agile software development: a clash of two cultures?*, volume 2, pages 497–498. ACM, 2010. Cited in page 17

Kruchten(2011) Philippe Kruchten. Contextualizing agile software development. *Journal of Software Maintenance and Evolution: Research and Practice*, page in press, 2011. doi: 10.1002/smr.572. Cited in page 99, 145

Ktata and Lévesque(2010) Oualid Ktata and Ghislain Lévesque. Designing and implementing a measurement program for scrum teams: what do agile developers really need and want? In *Proceedings of the Third C* Conference on Computer Science and Software Engineering*, C3S2E '10, pages 101–107, New York, NY, USA, 2010. ACM. URL http://doi.acm.org/10.1145/1822327.1822341. Cited in page 42

Larman and Basili(2003) C. Larman and V. R. Basili. Iterative and incremental development: A brief history. *IEEE Computer Society*, 36:47–56, 2003. Cited in page 12, 15, 16

Larsen and Larsen(2012) Diana Larsen and Willem Larsen. *Virtuosity. Creating High-performance in Learning Communities*. Leanpub, 2012. Cited in page 122

Lee and Xia(2010) Gwanhoo Lee and Weidong Xia. Toward agile: An integrated analysis of quantitative and qualitative field data. *Management Information Systems Quarterly*, 34(1):87–114, 2010. Cited in page 1, 15, 17, 72, 92

Leffingwell(2007) Dean Leffingwell. *Scaling Software Agility: Best Practices for Large Enterprises (The Agile Software Development Series)*. Addison-Wesley Professional, 2007. ISBN 0321458192. Cited in page xi, 83, 84

Levine and Choi(2004) J. M. Levine and H. S. Choi. *Team cognition: Understanding the factors that drive process and performance*, chapter Impact of personnel turnover on team performance and cognition, pages 153–176. Washington, DC: American Psychological Association, 2004. Cited in page 80, 90

Levine and Moreland(1990) J. M. Levine and R. L. Moreland. Progress in small group research. *Annual Review of Psychology*, 41(1):585–634, 1990. Cited in page 82

Lewis *et al.***(1991)** J.A. Lewis, S.M. Henry and D.G. Kafura. An empirical study of the object-oriented paradigm and software reuse. In *Proc. Conference on Object-Oriented Programming Systems, Languages and Applications*, pages 184—-196, 1991. Cited in page 32

Licorish *et al.***(2009)** Sherlock Licorish, Anne Philpott and Stephen G. MacDonell. Supporting agile team composition: A prototype tool for identifying personality (in)compatibilities. In *Proceedings of the 2009 ICSE Workshop on Cooperative and Human Aspects on Software Engineering*, CHASE '09, pages 66–73, Washington, DC, USA, 2009. IEEE Computer Society. ISBN 978-1-4244-3712-2. Cited in page 90, 94

Lim(1994) Wayne C. Lim. Effects of reuse on quality, productivity and economics. *IEEE Software*, pages 23–30, September 1994. Cited in page 32

Lincoln and Guba(1985) Yvonna .S. Lincoln and Egon G. Guba. *Naturalistic Inquiry*. Sage Publications, Newbury Park, CA, 1985. Cited in page 51, 95

Lindstrom and Jeffries(2004) Lowell Lindstrom and Ron Jeffries. Extreme programming and agile software development methodologies. *Information Systems Management*, 21(3):41–52, 2004. Cited in page 17

Lohan *et al.***(2010)** Garry Lohan, Kieran Conboy and Michael Lang. Beyond budgeting: A performance management model for software development teams. In Pekka Abrahamsson, Nilay Oza, Wil Aalst, John Mylopoulos, Michael Rosemann, Michael J. Shaw and Clemens Szyperski, editors, *Lean Enterprise Software and Systems*, volume 65 of *Lecture Notes in Business Information Processing*, pages 126–138. Springer Berlin Heidelberg, 2010. Cited in page 2, 43

Lu *et al.***(2011)** Yaobin Lu, Chunjie Xiang, Bin Wang and Xiaopeng Wang. What affects information systems development team performance? an exploratory study from the perspective of combined socio-technical theory and coordination theory. *Computers in Human Behavior*, 27(2): 811 – 822, 2011. Cited in page 71

Lycett *et al.***(2003)** Mark Lycett, Robert D. Macredie, Chaitali Patel and Ray J. Paul. Migrating agile methods to standardized development practice. *Computer*, 36(6):79–85, June 2003. ISSN 0018-9162. Cited in page 13

Lynch and Cross(1991) R. L. Lynch and K. F. Cross. *Measure up! The Essential Guide to Measuring Business Performance*. Mandarin, 1991. Cited in page 30

Lyytinen and Rose(2006) Kalle Lyytinen and Gregory M. Rose. Information system development agility as organizational learning. *EJIS*, 15(2):183–199, 2006. Cited in page 15, 16

MacCormack *et al.***(2003)** A MacCormack, C F Kemerer, M Cusumano and B Crandall. Trade-offs between productivity and quality in selecting software development practices. *IEEE Software*, 20(5):78–85, 2003. Cited in page 37

MacCormack *et al.***(2001)** Alan MacCormack, Roberto Verganti and Marco Iansiti. Developing products on "internet time": The anatomy of a flexible development process. *Management Science*, 47:133–150, January 2001. ISSN 0025-1909. Cited in page 17, 34, 37, 72

Malone and Crowston(1994) T. W. Malone and Kevin Crowston. The interdisciplinary study of coordination. *ACM Computing Surveys*, 26:87–119, March 1994. ISSN 0360-0300. Cited in page 83

Malone *et al.***(1993)** T.W. Malone, K. Crowston, Jintae Lee and B. Pentland. Tools for inventing organizations: toward a handbook of organizational processes. In *Proceedings of the Second Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises*, pages 72 –82, apr 1993. Cited in page 83

Marks *et al.***(2001)** M A Marks, J E Mathieu and S J Zaccaro. A temporally based framework and taxonomy of team processes. *Academy of Management Review*, 26(3):356–376, 2001. Cited in page 8, 38, 39, 71

Marks and Panzer(2004) Michelle A Marks and Frederick J Panzer. The influence of team monitoring on team processes and performance. *Human Performance*, 17(1):25–41, 2004. Cited in page 38, 39

Mathieu *et al.***(2000)** J E Mathieu, T S Heffner, G F Goodwin, E Salas and J A Cannon-Bowers. The influence of shared mental models on team process and performance. *Journal of Applied Psychology*, 85(2):273–283, 2000. Cited in page 38

Mathieu *et al.***(2008)** John Mathieu, M. Travis Maynard, Tammy Rapp and Lucy Gilson. Team effectiveness 1997-2007: A review of recent advancements and a glimpse into the future. *Journal of Management*, 34(3):410–476, 2008. Cited in page 71, 96

Maurer and Martel(2001) Frank Maurer and Sebastien Martel. On the productivity of agile software practices: an industrial case study. Technical report, University of Calgary, 2001. Cited in page 28, 29

Maxwell and Forselius(2000) K D Maxwell and Pekka Forselius. Benchmarking software development productivity. *IEEE Software*, 17(1):80–88, 2000. Cited in page 33, 34, 36, 37, 72, 91

Maxwell *et al.***(1996)** Katrina D. Maxwell, Luk Van Wassenhove and Soumitra Dutta. Software development productivity of european space, military, and industrial applications. *IEEE Transactions on Software Engineering*, 22:706–718, October 1996. Cited in page 32, 33, 34, 35, 36, 37

McAvoy and Butler(2007) John McAvoy and Tom Butler. The impact of the Abilene Paradox on double-loop learning in an agile team. *Information and Software Technology*, 49:552–563, June 2007. ISSN 0950-5849. Cited in page 17, 91

McHugh *et al.***(2011)** O. McHugh, K. Conboy and M. Lang. Using agile practices to influence motivation within it project teams. *Scandinavian Journal of Information Systems (Special Issue on IT Project Management)*, 23(2):85–110, 2011. Cited in page 88, 93

McKay and Marshall(2001) Judy McKay and Peter Marshall. The dual imperatives of action research. *Information Technology People*, 14(1):46–59, 2001. Cited in page 99

Melo *et al.***(2011)** C. Melo, D. S. Cruzes, F. Kon and R. Conradi. Agile team perceptions of productivity factors. In *Proceedings of the Agile 2011 (AGILE'11)*, pages 57–66, Salt Lake City, UT, USA, 2011. IEEE Computer Society. Cited in page 2

Melo *et al.***(2012)** Claudia de O. Melo, Célio Santana and Fabio Kon. Developers motivation in agile teams. In *36th EUROMICRO Conference on Software Engineering and Advanced Applications (SEAA)*, pages 376–383, Los Alamitos, CA, USA, 2012. IEEE Computer Society. Cited in page 8, 93, 110, 113

Menand(1997) L. Menand. *Pragmatism: a reader*. Vintage Series. Vintage Books, 1997. ISBN 9780679775447. Cited in page 4

Miles and Huberman(1994) Matthew B. Miles and A. M. Huberman. *Qualitative Data Analysis: An Expanded Sourcebook*, volume 2nd. Sage, 1994. Cited in page 71, 77, 78

Mintzberg(1987) Henry Mintzberg. Crafting strategy. *Harvard Business Review*, 65(4):66–75, 1987. Cited in page 30

Miranda and Bourque(2010) Eduardo Miranda and Pierre Bourque. Agile monitoring using the line of balance. *Journal of Systems and Software*, 83:1205–1215, July 2010. ISSN 0164-1212. Cited in page 42

Mishra and Mishra(2009) Deepti Mishra and Alok Mishra. Effective communication, collaboration, and coordination in extreme programming: Human-centric perspective in a small organization. *Human Factors and Ergonomics in Manufacturing & Service Industries*, 19(5):438–456, 2009. ISSN 1520-6564. Cited in page 92

Moe *et al.***(2009)** N. B. Moe, T. Dingsoyr and T. Dybå. Overcoming barriers to self-management in software teams. *IEEE Software*, 26(6):20–26, 2009. ISSN 0740-7459. Cited in page 92

Moe *et al.***(2010)** N. B. Moe, T. Dingsoyr and T. Dybå. A teamwork model for understanding an agile team: A case study of a scrum project. *Information and Software Technology*, 52(5):480 – 491, 2010. ISSN 0950-5849. doi: DOI:10.1016/j.infsof.2009.11.004. Cited in page 40, 73, 77, 92

Moe(2011) Nils Brede Moe. *From Improving Processes to Improving Practice. Software Process Improvement in Transition from Plan-driven to Change-driven Development*. PhD thesis, Norwegian University of Science and Technology, 2011. Cited in page 12, 13, 55

Moe *et al.***(2008)** Nils Brede Moe, Torgeir Dingsøyr and Tore Dybå. Understanding self-organizing teams in agile software development. In *Australian Software Engineering Conference*, pages 76–85, 2008. Cited in page 73, 108, 110, 113

Mohagheghi and Conradi(2007) Parastoo Mohagheghi and Reidar Conradi. Quality, productivity and economic benefits of software reuse: a review of industrial studies. *Empirical Software Engineering*, 12:471–516, October 2007. ISSN 1382-3256. Cited in page 95

Morasca(2001) Sandro Morasca. *Handbook of software engineering & knowledge engineering*, volume 1, chapter Software Measurement, pages 239–276. World Scientific Publishing, 2001. Cited in page 23, 24, 26

Morgeson *et al.***(2006)** Frederick P. Morgeson, Gina J. Medsker and Michael A. Campion. *Job and Team Design*, pages 428–457. John Wiley & Sons, Inc., 2006. Cited in page 114

Morisio *et al.***(1999)** M Morisio, D. Romano and C Moiso. Framework based software development: Investigating the learning effect. In *Proceedings of the 6th International Symposium on Software Metrics*, pages 260—-268, Boca Raton, FL, 1999. IEEE Computer Society Press. Cited in page 32

Moser *et al.***(2007)** Raimund Moser, Pekka Abrahamsson, Witold Pedrycz, Alberto Sillitti and Giancarlo Succi. A case study on the impact of refactoring on quality and productivity in an agile team. In *Balancing Agility and Formalism in Software Engineering, Second IFIP TC 2 Central and East European Conference on Software Engineering Techniques, CEE-SET 2007, Poznan, Poland*, pages 252–266. Springer, 2007. Cited in page 28

Mulder(1999) Martin Mulder. Case studies in performance improvement. *Advances in Developing Human Resources*, 1(1):83–94, 1999. Cited in page 74

Munro and Laiken(2003) Carolin Rekar Munro and Marilyn E Laiken. Developing and sustaining high performance teams managing difference through an action research approach. *OS Practitioner*, 35(4):62–67, 2003. Cited in page 55

Na *et al.***(2007)** Kwan-Sik Na, James T. Simpson, Xiaotong Li, Tushar Singh and Ki-Yoon Kim. Software development risk and project performance measurement: Evidence in korea. *Journal of Systems and Software*, 80(4):596–605, April 2007. Cited in page 24

Naumann and Jenkins(1982) J. D. Naumann and A. M. Jenkins. Prototyping: the new paradigm for systems development. *MIS Quarterly*, 6:29–44, 1982. Cited in page 12

Nerur *et al.***(2005)** S. Nerur, R. Mahapatra and G. Mangalaraj. Challenges of migrating to agile methodologies. *Communications of the ACM*, 48(5):72–78, 2005. Cited in page 11, 12, 13, 14

Nerur and Balijepally(2007) Sridhar Nerur and VenuGopal Balijepally. Theoretical reflections on agile development methodologies. *Commun. ACM*, 50:79–83, March 2007. ISSN 0001-0782. Cited in page 14, 72, 91

Nerur *et al.***(2010)** Sridhar Nerur, Alan Cannon, VenuGopal Balijepally and Philip Bond. Towards an understanding of the conceptual underpinnings of agile development methodologies. In Torgeir Dings/oyr, Tore Dybaa and Nils Brede Moe, editors, *Agile Software Development*, pages 15–29. Springer Berlin Heidelberg, 2010. Cited in page 11, 12

Neuman(2006) W. Lawrence Neuman. *Basics of social research: qualitative and quantitative approaches*. The Intersections Collection: Custom Resources in Sociology Series. Pearson-/Allyn and Bacon, 2006. ISBN 9780205484379. URL http://books.google.com.br/books?id=dUvZAAAAIAAJ. Cited in page xiii, 47

Parolia *et al.***(2007)** Neeraj Parolia, Stephen Goodman, Yuzhu Li and James J. Jiang. Mediators between coordination and is project performance. *Information & Management*, 44:635–645, October 2007. ISSN 0378-7206. Cited in page 93

Patton(1999) M. Q. Patton. Enhancing the quality and credibility of qualitative analysis. *Health services research*, 34(5 Pt 2):1189–1208, September 1999. ISSN 0017-9124. Cited in page 53, 54, 95

Peng *et al.***(2008)** David Xiaosong Peng, Roger G. Schroeder and Rachna Shah. Linking routines to operations capabilities: A new perspective. *Journal of Operations Management*, 26(6):730 – 748, 2008. Cited in page 17, 18

Petersen and Wohlin(2011) K. Petersen and C. Wohlin. Measuring the flow in lean software development. *Software-Practice & Experience*, 41(9):975–996, August 2011. Cited in page 41, 42

Petersen(2011) Kai Petersen. Measuring and predicting software productivity: A systematic map and review. *Information and Software Technology*, 53(4):317 – 343, 2011. Special section: Software Engineering track of the 24th Annual Symposium on Applied Computing - Software Engineering track of the 24th Annual Symposium on Applied Computing. Cited in page xiii, 2, 21, 23, 27, 28, 29, 40, 41, 43

Petersen(2012) Kai Petersen. A palette of lean indicators to detect waste in software maintenance: A case study. In *13th International Conference on Agile Processes in Software Engineering and Extreme Programming (XP'2012)*, pages 108–122, 2012. Cited in page 105

Pikkarainen *et al.***(2008)** M. Pikkarainen, J. Haikara, O. Salo, P. Abrahamsson and J. Still. The impact of agile practices on communication in software development. *Empirical Softw. Engg.*, 13: 303–337, June 2008. ISSN 1382-3256. Cited in page 1, 77

Pinsonneault and Kraemer(1993) Alain Pinsonneault and Kenneth L. Kraemer. Survey research methodology in management information systems: an assessment. *Journal of Management Information Systems - Special section: Strategic and competitive information systems*, 10(2):75–105, September 1993. ISSN 0742-1222. Cited in page 48

Poppendieck and Poppendieck(2003) Mary Poppendieck and Tom Poppendieck. *Lean Software Development: An Agile Toolkit*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2003. ISBN 0321150783. Cited in page 28, 29, 40, 41, 104, 106, 112

Poppendieck and Poppendieck(2006) Mary Poppendieck and Tom Poppendieck. *Implementing Lean Software Development: From Concept to Cash*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2006. Cited in page 40, 41, 112

Potok and Vouk(1998) T.E. Potok and M.A. Vouk. The effects of the business model on object-oriented software development productivity. *IBM Systems Journal*, 36(1):140–161, 1998. Cited in page 36

Pries-Heje and Commisso(2010) Jan Pries-Heje and Trine Hald Commisso. Improving team performance. In *Proceedings of the Information Systems Research In Scandinavia (IRIS)*, pages 1–12, 2010. Cited in page 2, 24, 43

Procaccino *et al.***(2005)** J. Drew Procaccino, June M. Verner, Katherine M. Shelfer and David Gefen. What do software practitioners really think about project success: an exploratory study. *Journal of Systems and Software*, 78:194–203, November 2005. ISSN 0164-1212. Cited in page 34, 72

Putnam and Myers(1996) L.H. Putnam and W. Myers. *Executive briefing: controlling software development*. IEEE Computer Society Press, 1996. ISBN 9780818674525. URL http://books.google.com.br/books?id=aoMhAQAAIAAJ. Cited in page 33

Qumer and Henderson-Sellers(2008) A. Qumer and B. Henderson-Sellers. An evaluation of the degree of agility in six agile methods and its applicability for method engineering. *Information and Software Technology*, 50(4):280–295, March 2008. ISSN 0950-5849. Cited in page 15, 16

Qumer and Henderson-Sellers(2006) Asif Qumer and Brian Henderson-Sellers. Crystallization of agility back to basics. In *ICSOFT (2)*, pages 121–126, 2006. Cited in page 16, 17

Rafii(1995) Farshad Rafii. How important is physical collocation to product development success? *Business Horizons*, 38(1):78–84, 1995. Cited in page 86, 92

Ramírez and Nembhard(2004) Y. W. Ramírez and D. A. Nembhard. Measuring knowledge worker productivity: A taxonomy. *Journal of Intellectual Capital*, 5(4):602–628, 2004. Cited in page 22, 23, 29, 76, 121

Rapoport(1970) R. Rapoport. Three dilemmas of action research. *Human Relations*, 23(6): 499–513, 1970. Cited in page 54

Rasch and Tosi(1992) Ronald H. Rasch and Henry L. Tosi. Factors affecting software developers' performance: an integrated approach. *MIS Quarterly*, 16:395–413, 1992. ISSN 0276-7783. Cited in page 1

Rodríguez *et al.***(2012)** Pilar Rodríguez, Jouni Markkula, Markku Oivo and Kimmo Turula. Survey on agile and lean usage in finnish software industry. In *ESEM*, pages 139–148, 2012. Cited in page 68, 69

Royce(1970) W. W. Royce. Managing the development of large software systems. In *Proceedings of WESCON*, pages 1–9, 1970. Also available in Proc. of ICSE 9, Computer Society Press, 1987. Cited in page 12

Runeson *et al.***(2012)** P. Runeson, M. Host, A. Rainer and B. Regnell. *Case Study Research in Software Engineering: Guidelines and Examples*. Wiley, 2012. Cited in page 79

Salas *et al.***(2004)** E. Salas, D. E. Sims and C. Klein. *Encyclopedia of applied psychology*, volume 1, chapter Cooperation and Teamwork at Work, pages 497–505. San Diego: Academic Press., 2004. Cited in page 38, 71

Salas(2005) Eduardo Salas. Is there a "big five" in teamwork? *Small Group Research*, 36(5): 555–599, 2005. Cited in page 71

Salas and Cannon-Bowers(1996) Eduardo Salas and Janis A. Cannon-Bowers. *Methods, Tools, and Strategies for Team Training*, pages 249–279. American Psychological Association, 1996. Cited in page 39

Salo(2006) Outi Salo. *Enabling Software Process Improvement in Agile Software Development Teams and Organisations*. PhD thesis, University of Oulu, 2006. Cited in page 55

Salo and Abrahamsson(2007) Outi Salo and Pekka Abrahamsson. An iterative improvement process for agile software development. *Software Process: Improvement and Practice*, 12(1):81–100, 2007. Cited in page 13, 14

Salo and Abrahamsson(2008) Outi Salo and Pekka Abrahamsson. Agile methods in european embedded software development organisations: a survey on the actual use and usefulness of extreme programming and scrum. *IET Software*, 2(1):58–64, 2008. Cited in page 68, 69

Sampaio *et al.***(2010)** Suzana Candido de Barros Sampaio, Emanuella Aleixo Barros, Gibeon Soares de Aquino Junior, Mauro Jose Carlos e Silva and Silvio Romero de Lemos Meira. A review of productivity factors and strategies on software development. *Software Engineering Advances, International Conference on*, 0:196–204, 2010. doi: http://doi.ieeecomputersociety.org/10.1109/ICSEA.2010.37. Cited in page 32

Sandberg *et al.***(2011)** Anna Sandberg, Lars Pareto and Thomas Arts. Agile collaborative research: Action principles for industry-academia collaboration. *IEEE Software*, 28(4):74–83, 2011. Cited in page 123

Sato *et al.***(2008)** D. T. Sato, H. Corbucci and M. V. Bravo. Coding dojo: An environment for learning and sharing agile practices. In *Proceedings Agile 2008 Conference*, pages 459–464. IEEE Computer Society, 2008. Cited in page 116, 117

Sawyer and Guinan(1998) S. Sawyer and P. Guinan. Software development: Processes and performance. *IBM Systems Journal*, 34(7):552–569, 1998. Cited in page 34, 35, 36, 37

Scacchi(1994) Walt Scacchi. Understanding software productivity. In W David Hurley, editor, *Software engineering and knowledge engineering: trends for the next decade*, volume 4, pages 273–314. World Scientific, 1994. Cited in page 24, 25

Schwaber(2007) C. Schwaber. The truth about agile processes. Forrester Research, Inc., 2007. Cited in page 11

Schwaber(2002) Ken Schwaber. The impact of agile processes on requirements engineering. In *Proceedings of the International Workshop on Time Constrained Requirements Engineering*, pages 12–16, September 2002. Cited in page 17

Schwaber and Beedle(2001) Ken Schwaber and Mike Beedle. *Agile Software Development with Scrum*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 1st edition, 2001. ISBN 0130676349. Cited in page 1, 65, 74

Seaman(1999) Carolyn B. Seaman. Qualitative methods in empirical studies of software engineering. *IEEE Trans. Software Eng.*, 25(4):557–572, 1999. Cited in page 47, 48

Selm and Jankowski(2006) M. Van Selm and N. Jankowski. Conducting online surveys. *Quality & Quantity*, 40:435–456, 2006. Cited in page 48, 49, 51, 61, 62

Senge(1990) Peter M Senge. The leader's new work: Building learning organizations. *Sloan Management Review*, 32(1):7–23, 1990. Cited in page 13

Sharp and Robinson(2004) Helen Sharp and Hugh Robinson. An ethnographic study of XP practice. *Empirical Software Engineering*, 9:353–375, December 2004. ISSN 1382-3256. Cited in page 72

Sharp and Robinson(2010) Helen Sharp and Hugh Robinson. Three cs of agile practice: collaboration, coordination and communication. In Torgeir Dingsøyr, Tore Dybå and Nils Brede Moe, editors, *Agile Software Development: Current Research and Future Directions*, pages 61–85. Springer, Berlin, 2010. Cited in page 92

Sharp *et al.***(2009)** Helen Sharp, Nathan Baddoo, Sarah Beecham, Tracy Hall and Hugh Robinson. Models of motivation in software engineering. *Information & Software Technology*, 51(1):219–233, 2009. Cited in page 8, 34, 35, 79, 86, 87, 93

Shull *et al.***(2010)** Forrest Shull, Grigori Melnik, Burak Turhan, Lucas Layman, Madeline Diep and Hakan Erdogmus. What do we know about test-driven development? *IEEE Software*, 27: 16–19, November 2010. ISSN 0740-7459. Cited in page 1

Siy and Mockus(1999) Harvey P. Siy and Audris Mockus. Measuring domain engineering effects on software change cost. In *Proceedings of the 6th International Symposium on Software Metrics*, pages 304—-311, Boca Raton, FL, 1999. IEEE Computer Society Press. Cited in page 32

Sjøberg *et al.***(2007)** D.I.K. Sjøberg, T. Dybå and M. Jørgensen. The future of empirical methods in software engineering research. In *Future of Software Engineering*, pages 358–378. IEEE, May 2007. Cited in page 47, 52, 55

Smith *et al.***(2001)** R.K. Smith, J.E. Hale and A.S. Parrish. An empirical study using task assignment patterns to improve the accuracy of software effort estimation. *IEEE Transactions on Software Engineering*, 27:264–271, 2001. Cited in page 34, 35, 36

Softex(2010) Softex. *Software and IT Services: The Brazilian Industry in Perspective*, volume 1. Observatório SOFTEX, Campinas-SP, Brazil, 2010. Cited in page 62, 63

Stagl *et al.***(2006)** Kevin C. Stagl, C. Shawn Burke, Eduardo Salas and Linda Pierce. *Understanding Adaptability: A Prerequisite for Effective Performance Within Complex Environments*, chapter Team Adaptation: Realizing Team Synergy. Advances in Human Performance & Cognitive Engineering Research. Elsevier JAI, 2006. Cited in page 120

Staron and Meding(2011a) Miroslaw Staron and Wilhelm Meding. Monitoring bottlenecks in agile and lean software development projects - a method and its industrial use. In *Proceedings of the 12th international conference on Product-focused software process improvement*, PROFES'11, pages 3–16, Berlin, Heidelberg, 2011a. Springer-Verlag. ISBN 978-3-642-21842-2. Cited in page 42

Staron and Meding(2011b) Miroslaw Staron and Wilhelm Meding. Factors determining long-term success of a measurement program: An industrial case study. *e-Informatica*, 5(1):7–23, 2011b. Cited in page 100

Stewart and Gosain(2006) Katherine J. Stewart and Sanjay Gosain. The moderating role of development stage in free/open source software project performance. *Software Process: Improvement and Practice*, 11(2):177–191, 2006. ISSN 1099-1670. Cited in page 71

Strode *et al.***(2011)** Diane E. Strode, Beverley Hope, Sid L. Huff and Sebastian Link. Coordination effectiveness in an agile software development context. In *Proceedings of the 15th Pacific Asia Conference on Information Systems (PACIS)*, pages 1–16, 2011. Cited in page 92

Sudhakar *et al.***(2011)** Goparaju Purna Sudhakar, Ayesha Farooq and Sanghamitra Patnaik. Soft factors affecting the performance of software development teams. *Team Performance Management*, 17:187–205(19), 2011. Cited in page 1, 24, 38, 43

Sue and Ritter(2007) V. M. Sue and L. A. Ritter. *Conducting Online Surveys*. Sage Publications, Inc, 2007. URL http://www.worldcat.org/isbn/141293754X. Cited in page 61

Susman and Evered(1978) Gerald I Susman and Roger D Evered. An assessment of the scientific merits of action research. *Administrative Science Quarterly*, 23(4):582–603, 1978. Cited in page 54, 55, 99

Tan *et al.***(2009)** Thomas Tan, Qi Li, Barry Boehm, Ye Yang, Mei He and Ramin Moazeni. Productivity trends in incremental and iterative software development. In *Proceedings of 3rd International Symposium on Empirical Software Engineering and Measurement*, (ESEM '09), pages 1–10, Washington, DC, USA, 2009. IEEE Computer Society. ISBN 978-1-4244-4842-5. doi: http://dx.doi.org/10.1109/ESEM.2009.5316044. Cited in page 32, 33, 34, 36, 37, 72, 91

Tang and Kishore(2010) Xiao Tang and Rajiv Kishore. The antecedents and consequences of agile practices: A multi-period empirical study of software teams in time-bound projects. In *Proceedings of the International Conference on Information Systems (ICIS) and International Research Workshop on IT Project Management*, Saint Louis, Missouri, USA, 2010. Cited in page 38, 71

Tangen(2005) Stefan Tangen. Demystifying productivity and performance. *International Journal of Productivity and Performance Management*, 54(1):34–46, 2005. ISSN 1741-0401. Cited in page xi, 21, 22, 28, 29

Teasley *et al.***(2000)** Stephanie Teasley, Lisa Covi, M. S. Krishnan and Judith S. Olson. How does radical collocation help a team succeed? In *Proceedings of the 2000 ACM conference on Computer supported cooperative work*, CSCW '00, pages 339–346, New York, NY, USA, 2000. ACM. ISBN 1-58113-222-0. Cited in page 36, 82, 86, 92

Tom *et al.***(2013)** Edith Tom, Aybuke Aurum and Richard Vidgen. An exploration of technical debt. *Journal of Systems and Software*, in press(0):–, 2013. Cited in page 110

Trendowicz and Münch(2009) Adam Trendowicz and Jürgen Münch. Factors influencing software development productivity - state-of-the-art and industrial experiences. *Advances in Computers*, 77:185–241, 2009. Cited in page 1, 21, 23, 28, 32, 33, 72

Tuckman(1965) Bruce W. Tuckman. Developmental sequence in small groups. *Psychological Bulletin*, 63(6):384–399, 1965. Cited in page 38, 72, 90

Tziner and Birati(1996) Aharon Tziner and Assa Birati. Assessing employee turnover costs: A revised approach. *Human Resource Management Review*, 6(2):113–122, 1996. Cited in page 80

Umarji and Emurian(2005) Medha Umarji and Henry H. Emurian. Acceptance issues in metrics program implementation. In *IEEE METRICS*, page 20, 2005. Cited in page 100

Utterback and Abernathy(1975) J.M. Utterback and W.J. Abernathy. *A Dynamic Model of Process and Product Innovation by Firms*. Center for Policy Alternatives at the Massachusetts Institute of Technology, 1975. Cited in page xi, 18, 100

van der Vegt *et al.***(2010)** Gerben S. van der Vegt, Stuart Bunderson and Ben Kuipers. Why turnover matters in self-managing work teams: Learning, social integration, and task flexibility. *Journal of Management*, 36(5):1168–1191, 2010. Cited in page 89, 90, 91, 96

Venkatesh and Davis(2000) Viswanath Venkatesh and Fred D Davis. A theoretical extension of the technology acceptance model: Four longitudinal field studies. *Management Science*, 46(2):186–204, 2000. URL http://mansci.journal.informs.org/cgi/doi/10.1287/mnsc.46.2.186.11926. Cited in page 100

Verner *et al.***(2009)** J.M. Verner, J. Sampson, V. Tosic, N.A.A. Bakar and B.A. Kitchenham. Guidelines for industrially-based multiple case studies in software engineering. In *Research Challenges in Information Science, 2009. RCIS 2009. Third International Conference on*, pages 313 –324, April 2009. doi: 10.1109/RCIS.2009.5089295. Cited in page 73

VersionOne(2008) VersionOne. 3th annual state of agile development. http://versionone.com/state_of_agile_development_survey/08, 2008. Cited in page 68

VersionOne(2009) VersionOne. 4th annual state of agile development. http://versionone.com/state_of_agile_development_survey/09, 2009. Cited in page 66, 68

VersionOne(2010) VersionOne. 5th annual state of agile development. http://versionone.com/state_of_agile_development_survey/10, 2010. Cited in page 40, 61, 66, 68, 69

VersionOne(2011) VersionOne. 6th annual state of agile development. http://www.versionone.com/pdf/2011_State_of_Agile_Development_Survey_Results.pdf, 2011. Cited in page 68

Vignali and Zundel(2003) Claudio Vignali and Mike Zundel. The marketing management process and heuristic devices: an action research investigation. *Marketing Intelligence Planning*, 21(4): 205–219, 2003. URL http://www.emeraldinsight.com/10.1108/02634500310480095. Cited in page xi, 57

Vosburgh *et al.***(1984)** J. Vosburgh, B. Curtis, R. Wolverton, B. Albert, H. Malec, S. Hoben and Y. Liu. Productivity factors and programming environments. In *Proceedings of the 7th international conference on Software engineering*, ICSE '84, pages 143–152, Piscataway, NJ, USA, 1984. IEEE Press. Cited in page 32, 33, 34, 36, 37

Wageman(2001) Ruth Wageman. How leaders foster self-managing team effectiveness: Design choices versus hands-on coaching. *Organization Science*, 12:559–577, September 2001. ISSN 1526-5455. Cited in page 72

Wagner and Ruhe(2008) S. Wagner and M. Ruhe. A structured review of productivity factors in software development. Technical Report Technical Report TUM-I0832, Institut für Informatik - Technische Universität München, 2008. Cited in page 1, 32

Wallace *et al.***(2004)** Linda Wallace, Mark Keil and Arun Rai. Understanding software project risk: a cluster analysis. *Information & Management*, 42:115–125, December 2004. ISSN 0378-7206. Cited in page 89

Wang and Conboy(2009) Xiaofeng Wang and Kieran Conboy. Understanding agility in software development through a complex adaptive systems perspective. In *ECIS*, pages 2182–2193, 2009. Cited in page 16, 17

West and Grant(2010) Dave West and Tom Grant. Agile development: Mainstream adoption has changed agility - trends in real-world adoption of agile methods. Technical report, Forrester Research, January 2010. Cited in page 11, 68

Whitworth and Biddle(2007) Elizabeth Whitworth and Robert Biddle. Motivation and cohesion in agile teams. In Giulio Concas, Ernesto Damiani, Marco Scotto and Giancarlo Succi, editors, *Agile Processes in Software Engineering and Extreme Programming*, volume 4536 of *Lecture Notes in Computer Science*, pages 62–69. Springer Berlin / Heidelberg, 2007. Cited in page 71

Williams *et al.***(2000)** Laurie Williams, Robert R. Kessler, Ward Cunningham and Ron Jeffries. Strengthening the case for pair programming. *IEEE Softw.*, 17:19–25, July 2000. ISSN 0740-7459. Cited in page 85

Wohlin *et al.***(2000)** Claes Wohlin, Per Runeson, Martin Höst, Magnus C. Ohlsson, Bjöorn Regnell and Anders Wesslén. *Experimentation in software engineering: an introduction*. Kluwer Academic Publishers, Norwell, MA, USA, 2000. ISBN 0-7923-8682-5. Cited in page 47

Yeatts and Hyten(1998) Dale E. Yeatts and Cloyd Hyten. *High-performing self-managed work teams: a comparison of theory to practice*. Sage Publications, Thousand Oaks, 1998. Cited in page 8, 39, 71, 72, 73, 91

Yin(2008) Robert K. Yin. *Case Study Research: Design and Methods (Applied Social Research Methods)*. Sage Publications, Inc, 4th edition, 2008. Cited in page xi, xiii, 4, 51, 53, 54, 77