RAFAELA MANTOVANI FONTANA

MATURITY IN AGILE SOFTWARE DEVELOPMENT

Doctoral dissertation submitted in fulfillment of the requirements for the degree of Doctor of Philosophy in Informatics in the Graduate Program in Informatics of the Pontifical Catholic University of Paraná, Brazil.

Curitiba
2015

RAFAELA MANTOVANI FONTANA

MATURITY IN AGILE SOFTWARE DEVELOPMENT

Doctoral dissertation submitted in fulfillment of the requirements for the degree of Doctor of Philosophy in Informatics in the Graduate Program in Informatics of the Pontifical Catholic University of Paraná, Brazil.

Major Concentration Field: Computer Science

Supervisor: Prof. Dr. Sheila Reinehr
Co-supervisor: Prof. Dr. Andreia Malucelli

Curitiba
2015

Fontana, Rafaela Mantovani
F679m Maturity in agile software development / Rafaela Mantovani Fontana ;
2015 supervisor: Sheila Reinehr ; co-supervisor: Andreia Malucelli. – 2015.
168 f. : il. ; 30 cm

Tese (doutorado) – Pontifícia Universidade Católica do Paraná, Curitiba, 2015
Bibliografia: f. 139-154

1. Engenharia de Software. 2. Desenvolvimento de software ágil. 3. Software – Controle de qualidade. I. Reinehr, Sheila. II. Malucelli, Andreia. III. Pontifícia Universidade Católica do Paraná. Programa de Pós-Graduação em Informática. IV. Título.

CDD 22. ed. – 005.1

Pontifícia Universidade Católica do Paraná
Escola Politécnica
Programa de Pós-Graduação em Informática

ATA DE DEFESA DE TESE DE DOUTORADO
PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA

ÁREA DE CONCENTRAÇÃO: CIÊNCIA DA COMPUTAÇÃO

DEFESA DE TESE DE DOUTORADO Nº 035/2015

Aos 7 dias de Outubro de 2015 realizou-se a sessão pública de Defesa da Tese de Doutorado intitulada **"Maturity In Agile Software Development"** apresentada pela aluna **Rafaela Mantovani Fontana** como requisito parcial para a obtenção do título de Doutor em Informática, perante uma Banca Examinadora composta pelos seguintes membros:

Prof.ª Dr.ª Sheila Reinehr
PUCPR (Orientadora _____ APROVADO
(*assinatura*) (*aprov/reprov.*)

Prof.ª Dr.ª Andreia Malucelli (Co-orientadora)
PUCPR _____ APROVADO

Prof. Dr. . Alcides Calsavara
PUCPR _____ APROVADO

Prof. Dr. Rafael Prikladnicki
PUCRS _____ APROVADA

Prof. Dr. Guilherme Horta Travassos
COPPE/UFRJ _____ APROVADA

Prof.ª Dr.ª Ana Liddy Cenni de Castro Magalhães
UFMG _____ APROVADA

Conforme as normas regimentais do PPGIa e da PUCPR, o trabalho apresentado foi considerado aprovado (*aprovado/reprovado*), segundo avaliação da maioria dos membros desta Banca Examinadora. Este resultado está condicionado ao cumprimento integral das solicitações da Banca Examinadora registradas no Livro de Defesas do programa.

Programa de Pós-Graduação em Informática PPGIa PUCPR

Prof.ª Dr.ª Andreia Malucelli
Coordenadora do Programa de Pós-Graduação em Informática.

Mãe e Pai, esta tese é dedicada a vocês.
Obrigada por tudo, sempre.

ACKNOWLEDGMENTS

I cannot express enough thanks to some very special people:

To my daughter and my son, for the smiles, our conversations and the joy they bring to my life.

To my husband, for his love and companionship.

To my sister and my brother, for the suggestions and several reviews of my papers, presentations, figures and rationales.

To my godmother, for being a special sort of family supervisor, and for her tireless help while reading my manuscript and making suggestions on methods, format, and whatever she could help me with.

To my supervisors, Sheila Reinher and Andreia Malucelli for their support, encouragement and the unmeasurable learning opportunities they offered me throughout this journey.

To my friends and my family, who were always around, helping me one way or another: my dear sister-in-law, my aunties and my cousin. To my friends: many thanks for our talks and your reviews!

To Professor Victor Meyer Jr. for the theories and insights he provided me with during his Complexity and Organizational Management course.

To the team leaders and company managers that gave me the opportunity to talk to their employees and learn about their processes, challenges and improvements.

To the practitioners and researchers who gave me useful feedback about my results: Alexandre Freire, Caio Cestari Silva, Daniela Cruzes, Greice Roman, Matheus Haddad, Morten Elvang, Peng Liu, Rafael Sabbagh, Sabrina Marczak, Tore Dybå, Torgeir Dyngsøyr and Xiaofeng Wang.

To the Pontifical Catholic University of Paraná, for awarding me the research grants I needed to pursue my doctoral degree.

“L’essentiel est invisible pour les yeux.”

Antoine de Saint-Exupéry in Le Petit Prince

ABSTRACT

Maturity models are used on a regular basis to guide improvements in software engineering processes. When agile software development teams rely on these models to mature their practices, sustaining agility is hindered at the highest maturity levels by the increasing focus placed on defining processes. A software process improvement guide for agile teams must keep focused on people and on sustaining agility, rather than on an extensive definition and control of processes. Current agile maturity models address this need, but they still lack agreement on what maturity means to agile teams. Moreover, they prescribe practices that teams should adopt, even though there is evidence that agile teams hardly ever accomplish their jobs by following prescribed practices. This dissertation, thus, takes the nature of agile software development teams into account in order to characterize maturity in agile software development. To accomplish this objective, the study had two specific goals: the first one was to define maturity in agile software development, and the second one was to identify the mechanisms that teams apply to mature in agility. The research was organized as a mixed-method approach, investigating quantitative and qualitative data at three main stages. The first stage – the exploratory one – addressed the first objective through a survey with agile practitioners. The second stage – the explanatory one – relied on a multiple-case study with agile teams to accomplish the second objective. The third stage sought to evaluate how well our objectives were met. The findings have shown that agile maturity means fostering subjective capabilities, such as collaboration, communication, commitment, care, sharing and self-organization. In the mechanism for maturing in agile, people play the central role. Moreover, it presents ambidexterity as a key ability to maturity, and does not prescribe practices, but rather describes outcomes that agile teams pursue to improve their working processes. These outcomes are accomplished in practices learning, in team conduct, in deliveries pace, in features disclosure, in the care with the software product, in customer relationship and in organizational support.

Keywords: agile software development, maturity, software process improvement.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1. INTRODUCTION

In the software engineering field, maturity models are used on a regular basis to guide improvements in software development processes. The main models used for this purpose are the Capability Maturity Model Integration for Development – CMMI-DEV (CMMI Product Team, 2010), and the international standards ISO/IEC 15504 (ISO/IEC, 2004) and ISO/IEC 12207 (ISO/IEC, 2008). In Brazil, the reference model is the MPS-SW, Brazilian Software Process Improvement Reference Model (SOFTEX, 2012b).

Each maturity model is founded on an underlying concept for maturity and the roadmap that the element (person, object or social system) follows to mature (KOHLEGGER; MAIER; THALMANN, 2009). In the current established maturity models for software development teams, maturity is achieved when continuous improvement takes place. To accomplish that, teams must define work processes, as well as standardize and quantitatively manage them (CMMI Product Team, 2010).

The implementation of CMMI-DEV guidelines have been recognized as a means to improve project performance and accomplish other benefits (JIANG et al., 2004; AGRAWAL; CHARI, 2007; SUBRAMANIAN; JIANG; KLEIN, 2007), but other approaches to help teams developing better software have arisen. One of them is *agile software development*. Contrasting with CMMI-DEV endeavors, the implementation of agile methods for software development values “individuals and interactions over processes and tools; working software over comprehensive documentation; customer collaboration over contract negotiation; and responding to change over following a plan” (BECK et al., 2001).

As soon as agile software development methods spread worldwide, researchers and practitioners started to implement both agile and CMMI-DEV simultaneously, to have the best of both worlds: agility to deliver to customers, with disciplined processes. Studies in the field have been reporting that CMMI-DEV, for example, should be used to help organizations institutionalize agile methods. They also point out that agile thinking guarantees that processes are implemented efficiently while responding to changes, and CMMI-DEV guarantees that all relevant processes are considered with

appropriate discipline (PAULK, 2001, ANDERSON, 2005; BAKER, 2006; SUTHERLAND; JAKOBSEN; JOHNSON, 2007; CAFFERY; PIKKARAINEN; RICHARDSON, 2008; JAKOBSEN; JOHNSON, 2008; COHAN; GLAZER, 2009; SPOELSTRA; IACOB; VAN SINDEREN, 2011; AL-TARAWNEH; ABDULLAH; ALI, 2011; LINA; DAN, 2012; LUKASIEWICZ; MILER, 2012). The benefits of this combination were accomplished and even recognized as a “magic potion” (SUTHERLAND; JAKOBSEN; JOHNSON, 2007).

However, when agile teams implement a software process improvement model to improve the way they work, such as CMMI-DEV, the extensive definition and the control of the process hinders sustaining agility at the highest maturity levels (PAULK, 2001; LUKASIEWICZ; MILER, 2012). Having recognized this limitation, researchers and practitioners have been proposing agile maturity models (SCHWEIGERT et al., 2012; ÖZCAN-TOP; DEMIRÖRS, 2013; LEPPÄNEN, 2013). These models provide improvement guidelines to agile software development teams that wish to keep focus on people and interaction over processes and tools, as stated by the Agile Manifesto (BECK et al., 2001). The highest maturity levels in these models are based either on the concept of project performance, or that of highly productive teams, or that of sustaining agility (as presented in CHAPTER 2).

We have identified two issues with these current agile maturity models. The first is that a variety of proposals are available, with different structures, focuses and underlying values – evidence that the field is still being defined. The second is that all of these models prescribe the practices and stages of adoption that should be followed by the teams – but such adoption is usually too context-dependent, and there is evidence that agile teams struggle to follow prescribed practices (SIDKY; ARTHUR; BOHNER, 2007; SCHWEIGERT et al., 2012; KETTUNEN, 2012; FONTANA; REINEHR; MALUCELLI, 2014).

Benefits and limitations of agile software development methods have already been identified (DYBÅ; DINGSØYR, 2008; MELO et al., 2013) and they seem to be replacing traditional[1] methods for software development (BUSTARD; WILKIE; GREER, 2013). A recent survey has shown that almost every software development organization has experienced agile practices worldwide (VERSION ONE, 2015) and

[1] Here and throughout this dissertation, we refer as “traditional” to an organization or practice “typically associated with a plan-driven approach to software development”. (as in WAARDENBURG; VLIET, 2013, p. 2154).

the interest in adopting agile methods is also increasing in Brazil (MELO et al., 2013). Although worldwide agile adoption has crossed the borders of collocated teams and small firms (VERSION ONE, 2015), the majority (about 68%) of the Brazilian companies that adopt agile methods have an Information Technology department whose size ranges between 1 and 50 employees (MELO et al., 2013). Considering that 1) agile methods are mainly suitable for small enterprises (DYBÅ; DINGSØYR, 2008) and that 2) Brazilian industrial context has around 80,000 software companies with less than nineteen employees (SOFTEX, 2012), there is still plenty of room for the adoption of agile methods by Brazilian companies.

In this industrial and research context, a characterization of maturity in agile software development could help defining agility, assist companies at the beginning of agile adoption, help organizations to recognize the implementation of agile methods, and also serve as a guide for improvement (FONTANA; REINEHR; MALUCELLI, 2014), as highlighted in Figure 1.

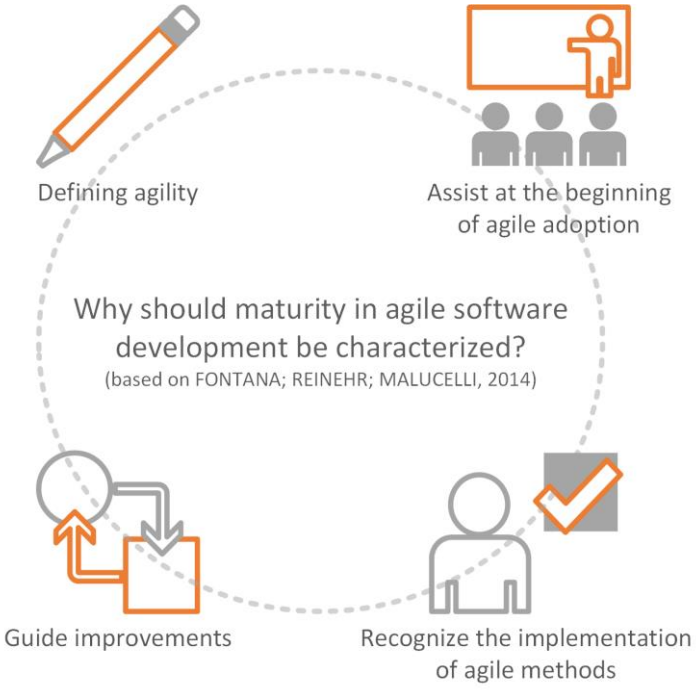


Figure 1 - Justification for maturity characterization in agile software development

The objective of this dissertation is, therefore, to *characterize maturity in agile software development*. Our purpose is to characterize this maturity because the nature of the element one wishes to aid in the maturing process must be considered in order to build useful guidelines for improvement (KOHLEGGER; MAIER; THALMANN,

2009). Therefore, we wish to understand how actual agile teams mature in order to describe this mechanism and build the basis for improvement guidelines.

Thus, to accomplish that, we defined two specific objectives: to *define* maturity in agile software development; and to identify the *mechanisms* that teams apply to mature in agile software development. Figure 2 shows our research questions, the general objective and the results we expect from the accomplishment of the specific objectives.

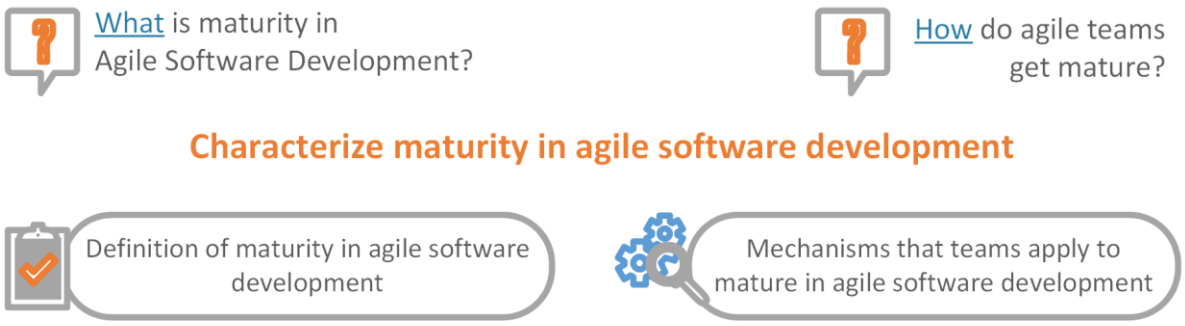


Figure 2 - Research questions and objectives

The findings of this study do not propose a CMMI-DEV-based model for guiding software process improvements with agile approaches. We show that agile maturity[2] means fostering subjective capabilities, such as collaboration, communication, commitment, care, sharing and self-organization. In the mechanism for maturing in agile, people play the central role. We show in our results that ambidexterity is as a key ability to maturity, and that specific practices should not be prescribed when guiding the maturing process. Rather, we present a framework that describes outcomes that agile teams pursue to improve their working processes. These outcomes are accomplished in practices learning, in team conduct, in deliveries pace, in features disclosure, in the care with the software product, in customer relationship and in organizational support.

This thesis is organized as follows. Chapter 2 presents the state-of-the-art in agile maturity models; Chapter 3 reviews the theoretical foundation we applied to analyze our data; and Chapter 4 shows the research approach. The results are described in Chapter 5 and, finally, Chapters 6 and 7 present our discussions and conclusions. Figure 3 summarizes the content of each chapter.

[2] Here and throughout this dissertation, we use the term “agile” to refer to “agile methods” or “agile approaches” for software development. Thus, “agile maturity” here means “maturity in agile software development”.

1

- Currently established maturity models are not applicable to sustain agility
- Agile maturity is still being defined
- Industrial context favors agile adoption
- We propose to characterize maturity in agile software development

2

- We present current agile maturity models
- They have different structures and different underlying concepts of maturity
- They still prescribe the practices that teams should accomplish

3

- We argue that agile teams, as complex adaptive systems, need management tools that foster the development of ambidextrous abilities
- We present our conceptual framework that considers agile teams experiment new practices to evolve to ambidexterity in a discontinuous way

4

- Our research is a behavioral-science study with a mixed-methods approach
- There were three main stages. The first aimed to define maturity, the second to identify the mechanisms to maturity, and the third to evaluate results
- Qualitative and quantitative data were collected and analyzed

5

- The definition of maturity in agile software development involves collaboration, communication, commitment, care, sharing and self-organization
- The mechanism teams apply to mature seem to be a discontinuous process of pursuit of progressive outcomes
- We present the Progressive Outcomes framework

6

- We discuss our definition of agile maturity, the inapplicability of stages or prescribed practices, the importance of management ambidexterity, the mature outcomes we identified on the teams and how we recognized agile continuous improvement
- We also present practical advice from the literature and discuss threats to validity of the research

7

- Based on the evaluation of our findings, we have accomplished the objective to characterize agile maturity
- The results are limited to the contexts from where the data was collected: Brazillian and inexperienced practitioners
- Further work should consider more empirical research to confirm the Progressive Outcomes in different contexts

Figure 3 - Thesis summary

CHAPTER 2. AGILE MATURITY MODELS

Chapter 1 contextualized the research gap, the motivation and the objectives for this study. Before presenting our theoretical foundation for the analysis of the results, this chapter provides an overview of currently published agile maturity models. Figure 4 contextualizes this chapter, as regards the previous and the next chapters.

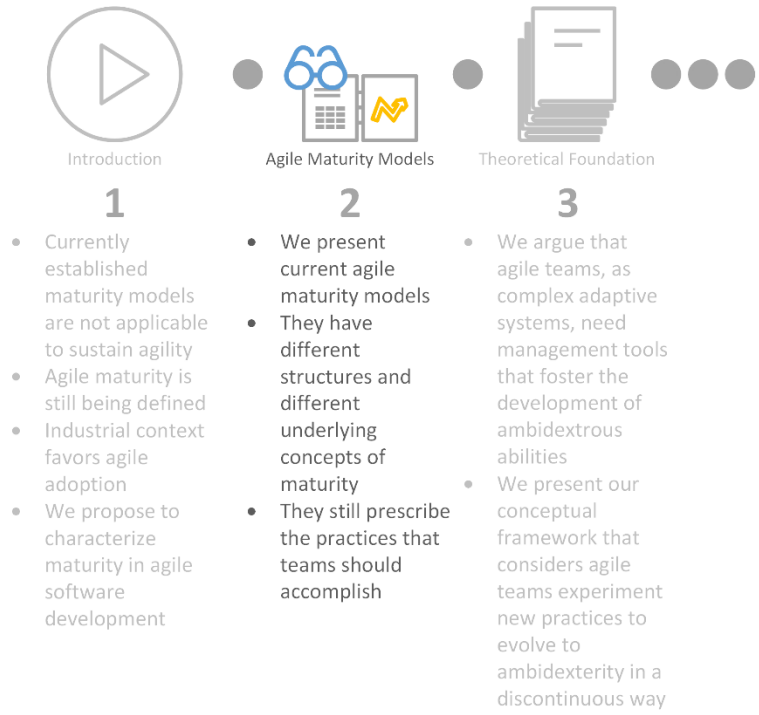


Figure 4 - Contextualization of Chapter 2

A number of agile software development methods have been proposed in recent years. Some of these methods emphasize software project management (Scrum and Internet-speed Development (ISD)); others support practices for software development (Agile Modeling, Extreme Programming (XP) and Pragmatic Programming), and there are those which emphasize both software project management and software development (Adaptive Software Development, Crystal Family, Dynamic Systems Development Model (DSDM) and Feature-driven Development (FDD)) (ABRAHAMSSON et al., 2003).

All of them share the values and principles disclosed in the Agile Manifesto (BECK et al., 2001) and suggest practices that embed agility in software development processes. This agility might be identified by characteristics such as being collaborative, being cooperative, being incremental, adaptability, self-organization, being iterative, constant testing, emergency, feedback incorporation, leanness, modularity, people orientation, reflection and introspection, small teams, time-boxing, and transparency (ABRANTES; TRAVASSOS, 2013).

The literature about maturity models and agile software development presents two main lines of research. The first one focuses on the combination of agile methods and CMMI-DEV. The second line presents models for maturing by sustaining agile values.

Studies that describe the combination of agile methods with CMMI-DEV respond to the needs of organizations that seek disciplined processes and agility (LUKASIEWICZ; MILER, 2012). Silva et al. (2015) perform an extensive literature review on these combinations. In summary, based on our analysis, such studies present the following reasons for combining such initiatives:

- agile processes are applicable to mature organizations (BUGLIONE, 2011);
- agile methods can be formalized by CMMI-DEV (BAKER, 2006);
- agile thinking guarantees that processes are implemented efficiently while responding to changes, and CMMI-DEV guarantees that all relevant processes are considered with appropriate discipline (JAKOBSEN; JOHNSON, 2008);
- CMMI-DEV should be used to help organizations institutionalize agile methods, and agile value is only obtained through discipline (SUTHERLAND; JAKOBSEN; JOHNSON, 2007);
- CMMI-DEV offers agile methods a systematic and quantitative approach to conducting projects by using well-known processes (COHAN; GLAZER, 2009);

These reasons highlight the benefits that companies may gain by combining agile and CMMI-DEV approaches. Nevertheless, considering the agile teams that wish to keep the characteristics of agile methods *and* mature in their practices, they cannot rely on CMMI-DEV and related approaches. Studies have already shown that agility cannot be sustained at the highest maturity levels, such as CMMI-DEV levels four and

five (PAULK, 2001; LUKASIEWICZ; MILER, 2012), because of the increasing definition of processes and control, which hinder agility.

Maturity models should consider the nature of the element they describe (KOHLEGGER; MAIER; THALMANN, 2009) and, for this reason, the definition of maturity in agile software development should not be based on processes, but rather on people (FONTANA et al., 2014b). The second line of research is comprised, thus, of studies that propose agile maturity models.

Agile maturity models have been proposed since the early years of agile methods. In 2001, Nawrocki et al. presented a four-level maturity model for Extreme Programming - XP (NAWROCKI; WALTER; WOJCIECHOWSKI, 2001). This model – as most models – is based on the CMMI-DEV structure. It is called XPMM (eXtreme Programming Maturity Model). The maturity levels are: “Not compliant at all”, “Initial”, “Advanced” and “Mature”. The evolution path starts with none of the Extreme Programming practices applied. It then evolves to customer relationship management and product quality assurance. Next, the team adopts pair programming and, at the highest maturity level, focus is placed on project performance.

Lui and Chan proposed another guide to the adoption of Extreme Programming (LUI; CHAN, 2005). Their aim was to guide the adoption of this agile method by inexperienced Chinese teams. The purpose is a four-stage roadmap. At each stage, the team incrementally implements more Extreme Programming practices. At the first stage, the team should implement testing, simple design, refactoring and coding standard. At the next stage, focus should be placed on continuous integration. At the third stage, the suggested practices are pair programming and collective ownership and, at the fourth and last stage, the team should adopt metaphor, forty-hour week, small release, on-site customer and planning game.

The proposal by Packlick (PACKLICK, 2007) is not focused specifically on Extreme Programming, as the previous models, but on agile methods in general. His proposal is called AGILE Roadmap and comprises five maturity levels: awareness, transformation, breakthrough, optimizing and mentoring. These levels are based on different stages of agile learning. The first level defines awareness of the goals; the second means that knowledge is put into practice; at the third level, agile practices are used to accomplish the goals and, at the fourth level, improvements are made continuously. The highest maturity in this model is the mentoring stage, in which

coaching is offered by high performance teams in order to share knowledge within the company.

Sidky et al. (2007) proposed a framework for adoption of agile methods (SIDKY; ARTHUR; BOHNER, 2007). The Sidky Agile Measurement Index (SAMI) defines five levels for agile adoption: collaborative, evolutionary, effective, adaptive, and encompassing. For each level, the author has defined which agile practices should be adopted. At the Collaborative level, focus is on collaboration; the Evolutionary level defines practices for continuous delivery; at the Effective level, focus is on increasing efficiency; at the next level, Adaptive, practices focus on responding to changes and, at the last level, Encompassing, agility is embedded into the organizational culture.

Also focusing on adoption of agile methods, the framework by Qumer and Henderson-Sellers (2008) defines the Agile Adoption and Improvement Model (AAIM) (QUMER; HENDERSON-SELLERS, 2008). This model defines six stages for the team to improve in agility. The team starts in the Agile Infancy, focusing on basic agile properties; then, it moves on to the Agile Initial stage, whose focus is to enable communication and collaboration. The next stage, Agile Realization, is represented by the use of executable artifacts and minimal documentation. The fourth stage, the Agile Value, focuses on valuing people although without ignoring tools and processes. The last two stages, Agile Smart and Agile Progress, focus on learning and on sustaining agility, respectively.

The model presented by Patel and Ramachandran (2009) has a similar structure to that of CMMI-DEV. They defined five stages for growing in maturity: Initial, Explored, Defined, Improved and Sustained. The highest maturity team, at the Sustained level, continuously improves software development process, manages uncertainties, tunes project performance and prevents defects on the software (PATEL; RAMACHANDRAN, 2009).

An empirical model, based on an experience in a single company, was introduced by Benefield (2010). He identified five levels that define to which extent a set of agile practices are adopted by the team: 1) Emergent Engineering Best Practices; 2) Continuous Practices at Component Level; 3) Cross Component Continuous Integration; 4) Cross Journey Continuous Integration; and 5) On Demand Just in Time Releases. At the highest level, teams are highly productive and new products are quickly delivered through a service-oriented architecture (BENEFIELD, 2010).

The proposal by Yin et al. (2011) is the Scrum Maturity Model, whose objective is to aid the adoption of Scrum, with a focus on customer relationship. The model has five levels. The Initial level is characterized by the absence of goals for process improvement. At the second level, Managed, the Scrum practices are more structured and, at the third level, focus is placed on the relationship with customers. At the forth level, Quantitatively Managed, the team uses metrics and manages project performance. The highest maturity level – Optimizing – focuses on performance management (YIN; FIGUEIREDO; SILVA, 2011).

A more recent approach for agility assessment has been proposed by Özcan-Top and Demirörs (2014), the Software Agility Assessment Model – AgilityMOD. Their model comprises two dimensions: Agility dimension and Aspect dimension. They named as “aspects” a set of agile processes and practices integrated under abstract definitions. Thus, aspects are classified into the categories Exploration (capturing customer needs and requirements-related activities), Construction (architecture and coding-related activities), Transition (deployment-related activities), Management (planning and monitoring-related activities) and Culture (environmental conditions and people behavior).

Each aspect attribute is related to agile principles and is defined as: Performing Aspect Practices, in the first level; Simple and Iterative, in the second level; Technically Excellent, in the third level; and Learning in the fourth level. The agility of one aspect is evaluated in a four-point scale: Not implemented, Ad-hoc, Lean and Effective, which are the agility levels, i.e. the Agility dimension.

Özcan-Top and Demirörs (2014)’s model is in early stages of evaluation, as the authors presented a single assessment based on the AgilityMOD. Their conclusion was that some improvements are needed considering that there are redundancies, missing practices, and excess of practices (ÖZCAN-TOP; DEMIRÖRS, 2014).

We analyzed each of these models according to the directions presented by Maier et al. (2012, p. 149) for planning and developing maturity models. The result of our analysis is shown in Table 1. The columns of the table present the following data:

- Audience: definition of the expected users of the model;
- Aim: the model may be applied to either one of two aims – analysis or benchmarking. The first aim helps determine the necessary improvements, while the second presents best practices for comparison by other organizations;

- Scope: defines if the model is generic or domain-specific;
- Success criteria: Maier et al. (2012) suggest that models must have criteria to define if the application was successful. They suggest evaluating usability and usefulness;
- Process areas: according to Maier et al. (2012), this is one of the most important aspects in maturity models because process areas must be “mutually exclusive and collectively exhaustive” (MAIER; MOUTRIE; CLARKSON, 2012, p. 150). Process areas uncover the conceptual foundation used in the development of the model;
- Maturity levels: Maturity levels must be based on a rationale that, according to Maier et al. (2012), may be: existence and adherence to a structured process, modification of organizational structure, emphasis on people, or emphasis on learning;
- Cell texts: The model must provide the characteristics at the intersection of process areas and maturity levels;
- Administration mechanism: definition of the mechanisms used for applying the model, that is, to conduct the assessment.

As one can realize, most of them are based on agile practices in general, and focus mostly on analytic (rather than benchmarking) implementation, and define four to six maturity levels mainly based on existence and adherence to a structured process, probably because of the influence of CMMI-DEV.

To provide a view of the concept of highest maturity underlying the models, we also performed an analysis of the highest maturity level for each model. Authors present quite different views on maturity. Although details are very distinct, the agile highest maturity has been defined under three main concepts: project performance, sustaining agility and highly productive teams. The first concept was proposed by Nawrocki et al. (2001), Patel and Ramachandran (2009) and Yin et al. (2011). The second is supported by Sidky et al. (2007) and Qumer and Henderson-Sellers (2008). Finally, the third was introduced by Packlick (2007) and Benefield (2010). These two are the only benchmark models that present good experiences at a single organization. The model proposed by Özcan-Top and Demirörs (2014) defines high-maturity as both high-performance teams and sustaining agility. The model by Lui and Chan (2005) did not fit into these concepts because the highest level only presents new XP practices to be adopted. Our analysis is shown in Table 2 .

The nine agile maturity models present different foundations, different aims and different structures. Even the concept of high-maturity behavior differs among them. It confirms conclusions presented in three published assessments of agile maturity models (SCHWEIGERT et al., 2012; LEPPÄNEN, 2013 and ÖZCAN-TOP; DEMIRÖRS, 2013).

In addition to that, we observed that the validation of the models is still at early stages and, for this reason, the quality of the current agile maturity models is still not completely ensured, as also observed by Leppänen (2013) and Özcan-top and Demirörs (2013). There is therefore still plenty of research to be conducted in the field. Leppänen (2013) noted that there is a clear need for more empirical studies and added some requirements for agile maturity models: they must be established over a strong conceptual basis, must be derived from successful agile development, and must present a well-defined structure.

Despite the fact that the agile community demonstrates interest in agile maturity models, some studies have shown that agilists do not feel comfortable about following models. Sidky et al. (2007), for example, realized that practitioners did not agree with the distribution of agile practices into levels in the adoption model they proposed. Similarly, Schweigert (2012), surveyed practitioners and found out that they do not believe in prescriptive models (they would prefer descriptive ones). Likewise, Kettunen (2012), when evaluating a matrix for measuring agility, found out that some respondents doubted the usefulness of assessing their agility in software development. Our survey with agile practitioners came to a similar conclusion (FONTANA; REINEHR; MALUCELLI, 2014). Figure 5 summarizes the limitations in current agile maturity models.

Table 1 - Analysis of the structure of agile maturity models

	Audience	Aim	Scope	Success criteria	Process areas	Maturity levels	Cell texts	Administration mechanism
NAWROCKI; WALTER; WOJCIECHOWSKI, 2001	Organizations implementing XP	Analytic	Only for XP	No evidence	Yes, based on intersections between XP and CMM	4 levels, based on existence an adherence to a structured process	Yes, defined through practices	Partial. Concludes that the assessment is subjective
LUI; CHAN, 2005	Inexperienced XP teams	Analytic	Only for XP	No evidence	Yes, based on visual data mining of XP practices	4 levels, based on learning	No evidence	No evidence
PACKLICK, 2007	Agile teams at Sarbre Airline Solutions	Benchmarking	Agile in general	Yes. Defined acceptance criteria for goals areas	Yes, based on goals	5 levels, based on people	Yes, defined through user stories	Yes, presents how they implemented it in the company
SIDKY; ARTHUR; BOHNER, 2007	Agile teams	Analytic	Agile in general	Yes. Defined project and organizational assessment	Yes, based on Agile Principles	5 levels, based on existence and adherence to a structured process	Yes, defined through practices	Yes, defines a four-stage process for agile adoption
QUMER; HENDERSON-SELLERS, 2008	Agile teams	Analytic	Agile in general	No evidence	No evidence	6 levels, based on existence and adherence to a structured process	No evidence	Partial. Describes two implementation cases
PATEL; RAMACHANDRAN, 2009	Agile teams	Analytic	Agile in general	Yes. Defined the assessment process	Yes, based on Agile Practices	5 levels, based on existence and adherence to a structured process	No evidence	Yes, exemplifies how to perform the assessment
BENEFIELD, 2010	XP teams at British Telecom	Benchmarking	Only for XP	Yes. Defined the assessment method	No, but defines 7 dimensions for assessment	5 levels, based on existence and adherence to a structured process	No evidence	Yes, exemplifies how to perform the assessment
YIN; FIGUEIREDO; SILVA, 2011	Scrum teams	Analytic	Only for Scrum	Yes. Defined metrics to evaluate implementation	Yes, named as “Goals”	5 levels, based on existence and adherence to a structured process	Defines practices for each goal, but they were not presented in the paper	Partial. Briefly describes six appraisals
ÖZCAN-TOP; DEMIRÖRS, 2014	Organizations implementing agile	Analytic	Agile in general	Yes. Partially present the assessment method	Yes, named as generic and aspect practices	4 levels, based on existence and adherence to a process (practices)	No evidence	Yes, exemplifies how to perform the assessment

Table 2 - The highest maturity levels in agile maturity models

Author	Highest Maturity is at	Main concept
NAWROCKI; WALTER; WOJCIECHOWSKI, 2001	Level 4 – Mature. Focus at this level is placed on customer and developer satisfaction. The results of the team are also considered. The only Process Area is Project Performance.	Project performance
LUI; CHAN, 2005	Stage 4. The team is focused on implementation of specific XP practices: metaphor, 40-hour week, small release, on-site customer and planning game.	-
PACKLICK, 2007	Level 5 – Mentoring. Stimulation of learning across the organization because high performance teams coach other teams according to specific goals.	Highly productive teams
SIDKY; ARTHUR; BOHNER, 2007	Level 5 – Encompassing. Focus is on sustaining agility. Agile practices at this level comprise low processes, agile project estimation, agile physical setup, TDD, experienced teams and frequent face-to-face communication.	Sustaining agility
QUMER; HENDERSON-SELLERS, 2008	Level 6 – Lean Production, Keep Agile. At this level, the work is focused on quality production, minimal possible resources and keeping agility.	Sustaining agility
PATEL; RAMACHANDRAN, 2009	Level 5 – Mature. There is continuous improvement based on quantitative data. The goals at this level are context improvement, uncertainty management, tuning project performance and defect prevention.	Project performance
BENEFIELD, 2010	Level 5 – On Demand Just in Time Release. High-level engineering practices are spread across the teams, which are highly productive and able to quickly assemble new products.	Highly productive teams
YIN; FIGUEIREDO; SILVA, 2011	Level 5 – Optimizing. The teams at these levels have experienced practitioners focused om continuous improvement. The only goal at this level is Performance Management.	Project performance
ÖZCAN-TOP; DEMIRÖRS, 2014	Level 4 – Effective. Agile engineering methods and tools are used to improve productivity (technical excellence) and there is a purpose for organizational learning and improvement (learning).	Highly productive teams and sustaining agility

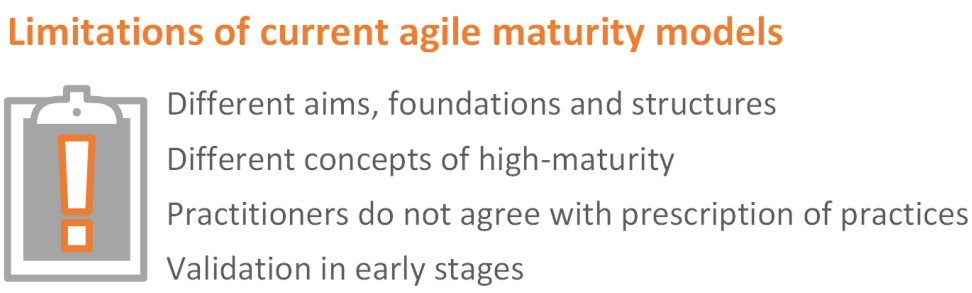


Figure 5 - Limitations of current agile maturity models

Despite the importance of people and their interaction in agile software development, only Packlick (2007)'s model places emphasis on people, i.e. the team. His model explains the maturing process based on team learning, contrasting with one of the main guidelines for team maturity in software engineering – the Team Software Process (TSP), created by CMMI's author, Watts Humphrey (HUMPHREY et al.,

2010). The philosophy in this model is similar to that of CMMI-DEV, which considers definition and standardization of processes as essential for maturing. According to the TSP guide, a team starts in a forming stage, where the team comes together as a working group and team members are highly dependent on their leader. This team then moves on to a storming stage where members start to vie for position among themselves and conflicts may occur. As the team evolves, it goes to a norming stage, in which team members reach consensus and trust grows. The last stage is the performing stage: the team works as a unit and energy is channeled into performing tasks (HUMPHREY et al., 2010).

This evolvement path is based on Tuckman's model for team development (TUCKMAN, 1965). This model assumes a linear path for evolvement and has currently been considered as unsuitable for self-managing teams (KUIPERS; STOKER, 2009), as agile teams are supposed to be. Instead, self-managed work teams have shown evidence of non-linear evolvement through three team processes that occur simultaneously. The first is “task management”, in which team members develop capabilities to manage responsibilities and control. The second is “internal relations”, in which the team deals with internal cooperative issues. Finally, the third is “external relations and improvement”, in which the team deals with its relationship with other teams, customers and suppliers. These teams evolve to greater self-management and, thereby, increased performance and enhanced quality of working life (KUIPERS; STOKER, 2009).

This particular feature of agile teams – allowing emergence and non-linearity on the expense of processes definition and control – bring to light the relation agile teams have with complex adaptive systems theory (HODA; NOBLE; MARSHALL, 2012; VIDGEN; WANG, 2009). This relation provides the basis for our theoretical foundation and the conceptual framework we used to analyze our data, presented in next chapter.

CHAPTER 3. THEORETICAL FOUNDATION

We observed, in the previous chapter, that agile methods lack a consolidated maturity model, and even a consolidated definition for maturity. For this reason, this chapter reviews the Complex Adaptive Systems theory in order to build the foundation required for understanding maturity in agile software development. Figure 6 shows our main conclusions in this chapter and the conclusions from the previous and the next chapters.

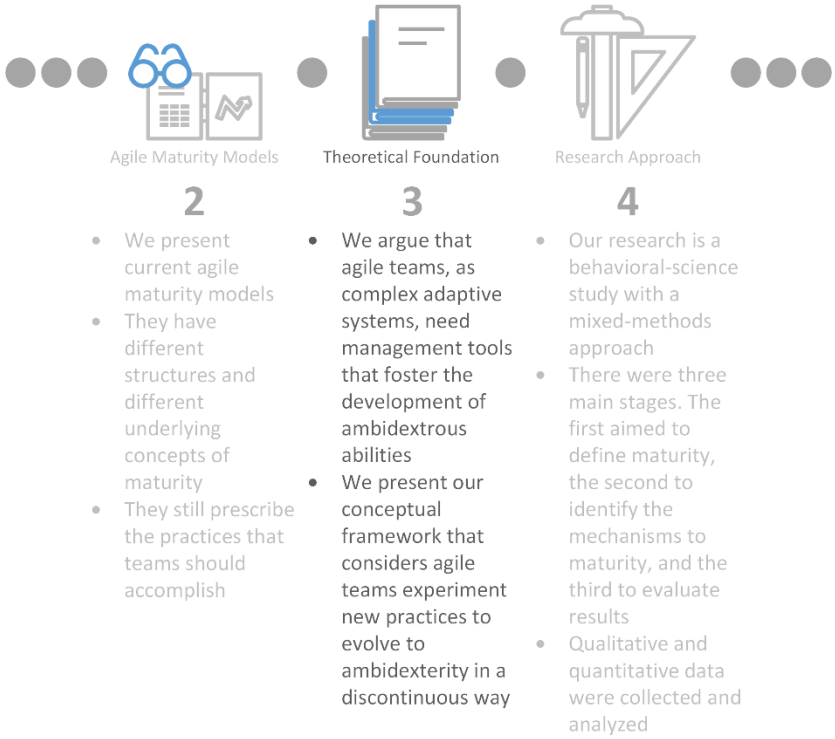


Figure 6 - Contextualization of Chapter 3

Agile software development teams are complex adaptive systems, as a number of researchers have already reported (AUGUSTINE et al., 2005; WHITWORTH; BIDDLE, 2007; VIDGEN; WANG, 2009; HODA; NOBLE; MARSHALL, 2012; POWER, 2014). Among other theories that could be used to explain agile teams dynamics – for instance knowledge management, personality, organizational learning, social facilitation and others (DINGSØYR et al., 2012) – we acknowledge that a complexity theoretical approach is in accordance with an emergent approach in social sciences: considering social systems as complex adaptive systems.

In organizational studies, it implies that companies show emergent behaviors that are neither static nor random – but “difficult to describe” (MAGUIRE; ALLEN; MCKELVEY, 2011). Although some researchers see complexity more as a metaphor or an analogy to human systems (MITLETON-KELLY, 2003), complexity theories are now becoming a major revolution in thinking. This theory prompts us to rethink the character of human intervention in the social and natural world (TSOUKAS, 2005).

This chapter describes the complex adaptive systems theory and the implications that this theory brings when it comes to understanding organizations. We also show how management strategies are changing in order to deal with complexity; after that, we discuss organizational ambidexterity. The chapter ends with our conceptual framework, shown in Figure 7, which consolidates the topics from theory that will be used in this dissertation.

3.1 Complex adaptive systems

> *“Complex adaptive systems consist of a number of components, or agents, that interact with each other according to a set of rules that require them to examine and respond to each other’s behavior in order to improve their behavior and thus the behavior of the system they comprise.” (STACEY, 1996, p.10).*

> *“When a system contains agents or populations that seek to adapt, we will use the term Complex Adaptive System.” (AXELROD; COHEN, 2000, p. 7)*

These two concepts of complex adaptive systems reveal that we are essentially referring to agents that interact to adapt. Properties and dynamics of such systems have been observed in a number of social groups – in nature and society – and the effects interventions generates in these systems have specific dynamics that might enrich our understanding about organizations. Before showing how it has been applied to organizations, we will present the underlying concepts of deterministic and adaptive feedback networks (STACEY, 1996). Studies of these kinds of feedback networks

provide useful reflections on how chemical and biological systems create complex behavior.

Deterministic feedback networks are a group of agents whose behaviors are determined by a common schema. A system with a pendulum or the weather are examples cited by Stacey (1996). Key concepts that describe these systems are rooted on the mathematical chaos theory and on the chemical dissipative structures theory.

The chaos theory states that systems, in their trajectories, are attracted to a point, which is called attractor. “An attractor is a pattern of behavior into which a system ultimately settles in the absence of outside disturbances” (STACEY, 1996, p. 54). Attractors might show different types of behavior: being in stable equilibrium, being unstable or being a strange attractor. Stable and unstable attractors present behaviors that never change. Our interest lies in the strange attractor, which shows stability of limits and shape, but it is impossible to predict the archetypes of such systems. They are unstable and stable at the same time, and this is the only state where the system is capable of novelty (STACEY, 1996).

The dissipative structures theory complements this view by explaining how systems use disorder to generate new order. This behavior was observed in chemical systems that are changeable only when they are pushed far from equilibrium. Researchers in this field observed that, in a dissipative system, self-organization happens after the system suffers a disorder by dissipating energy and information from the environment. At the same time, the system has a structure that allows agents to self-organize. This is a bottom-up process of change in which the pattern produced by self-organization cannot be reduced by the agent’s behavior (STACEY, 1996). The issue with such deterministic networks is that, although complex dynamics appear, individual agents – e.g. the pendulum, the wind in the weather or the chemical components – do not learn.

Another type of network, which shows similar dynamics, is the adaptive feedback network. According to Stacey (1996), in this kind of network, the behavior of the agents is determined by a common schema consisting of a few rules. In contrast to deterministic networks, the purpose of these systems is to perform a particular task and they learn it in a simple single-loop manner. It means that they simply adjust their behavior according to changes in the environment. Because agents seek to adapt, as shown in the concepts we presented in the beginning of the chapter, we characterize

this network as a complex adaptive system. Examples of such systems are a flock of birds and groups of ants.

Despite the difference in the behavior of individual agents, complex adaptive systems have quite the same properties as deterministic feedback networks: attractors, in which long-term evolution is not predictable, but short-term is; and space for novelty in the paradoxical state of stability and instability. Such dynamics have been simulated by computer programs, such as cellular automata and genetic algorithms (STACEY, 1996).

Knowledge about deterministic and adaptive feedback networks properties, as we briefly presented here, is relevant because human systems show similar structure and dynamics to those of other complex adaptive systems. However, Stacey (1996) emphasizes the difference: in human systems, agents' responses are driven by paradoxical emotions and feelings; agents share a common purpose, but also develop their own individual purposes; some agents are more powerful and use persuasion for others to follow them; and agents can adopt the role of observer and think systematically. For example, humans can deliberately create situations to push the systems to a state far from equilibrium and, in addition to that, provide support for a new order to be established (MITLETON-KELLY, 2003).

Mitleton-Kelly (2003) explains that complex behavior in human systems emerges, then, from the relationship, interactions and interconnectivity of the elements. Complex adaptive systems dynamics are still observed and, in addition to that, a decision or action of an element (or group of elements) in the system may affect other related elements and systems. The impact will not be the same for all of them and will depend on the history and on the structure of the system. Regardless of being aware or not, we all intervene in complex adaptive systems (AXELROD; COHEN, 2000).

3.2 Organizations as complex adaptive systems

According to Tsoukas (2005), many mainstream theories that explain organizational behavior follow a "Newtonian Style" of thinking. These theories see world as an idealized construction, an abstract model; for example, when considering that decision-making is a rational exercise that translates managerial talk into decisions and actions. When abstractions are put aside and complexity is brought into light, we see that executives, when making these decisions, are subject to a limited knowledge of the system they are part of – which is unavoidable (CILLIERS, 2002) – and decisions

are made based on the purposes of the actors involved and on the relations they are embedded in (TSOUKAS, 2005, p. 215; MARCH, 1988).

Organizations, as human systems, show two types of interactions between agents. These interactions may be explicitly stated – the legitimate network – or created spontaneously – the shadow network. The former consists of the links between agents that are formally and intentionally established by management or by principles that are widely accepted, e.g., a shared culture. The latter consists of informal social and political links where local rules are developed. Here, there are not just flows of information, energy and action, but also flows of emotion, friendship, trust and other qualities (STACEY, 1996).

These informal interconnections in the shadow network imply that an agent's decision or action may affect related agents in ways that cannot be predicted (MITLETON-KELLY, 2003). People in organizations "hardly ever accomplish their joint action on an ordinary day-to-day basis in a way that is entirely determined by the designed system they operate within" (STACEY; GRIFFIN; SHAW, 2000, p. 186). The networks that define an organization are not those "charts […] printed down on an organization meeting room", but the ones that emerge from informal interactions (HIDALGO, 2011, p. 557).

In this context, the concept of self-organization, which was first observed in deterministic feedback networks, becomes a valuable property of complex adaptive systems and is explicitly recognized in agile teams, as expressed in the Agile Manifesto (BECK et al., 2001). Self-organization happens when systems decide how to accomplish a given task. It allows the emergence of new ideas, relationships and organizational shapes, as well as organizational learning (MITLETON-KELLY, 2003). The actual practices the system adopts to accomplish the tasks emerge naturally from praxis (CAMPBELL-HUNT, 2007), as long as this emergence is allowed by the structure in the system.

Over time, complex systems evolve by combining stability and flexibility through processes of continuous experimentation (DEROSA; MCCAUGHIN, 2007). There is no such a linear path for growth in these systems: they evolve their structure and dynamics in a discontinuous manner (EIJNATTEN, 2003), like the strange attractors concept presented earlier.

Summing up, this rationale shows that organizations do have shadow networks, self-organization is a means for creating innovation, and evolution is supported by

stability and flexibility simultaneously. By acknowledging that, we understand that managers' perceptions are, in fact, highly inaccurate, and this suggests they need to apply tools that do not take the existence of accuracy for granted when guiding organizations towards achieving their goals (WEICK; SUTCLIFFE; OBSTFELD, 2005).

3.3 Management in complex organizations

When Campbell-Hunt (2007) states that order is emergent in a complex system, management must be aware that the consequences of managerial action may be difficult – or even impossible – to predict (AXELROD; COHEN, 2000). Axelrod and Cohen (2000), in response to the inapplicability of controlling the system, suggest that we should "harness complexity". It means "living with it, and even taking advantage of it, rather than trying to ignore or even eliminate it" (AXELROD; COHEN, 2000, p. 9).

Snowden and Boone (2007) agree by stating that "wise executives tailor their approach to fit the complexity of the circumstances they face" (SNOWDEN; BOONE, 2007, p. 1). The authors introduced the Cynefin framework to characterize managerial actions required for different contexts. Simple and complicated contexts have perceptible cause-and-effect relations. Thus, right answers can be based on facts. On the other hand, complex and chaotic contexts are unordered and the cause-and-effect relationship is not apparent. Thus, action should be based on emerging patterns.

Most situations and decisions in organizations are complex and events can be understood only in retrospect. This is the reason why Snowden and Boone (2007) suggest managers should probe, sense and, then, respond – and do not fall into the temptation of resorting to the traditional management style of command and control (MAGUIRE; ALLEN; MCKELVEY, 2011). For this reason, these authors suggest the following managerial tools: opening up the discussion, setting the barriers with simple rules, stimulating attractors, encouraging dissent and diversity; and managing start conditions and monitoring for emergence.

Likewise, McDaniel Jr. (2007) states that traditional command, control and planning are not effective in complex settings: "Commands are subject to nonlinear interdependencies among agents. Control is sensitive to initial conditions and self-organization of work. Planning changes the world and by that very fact, can cause co-evolution and cannot be predicted". (MCDANIEL JR., 2007, p. 27). His suggestion is that managerial tasks should, thus, be focused on sensemaking (WEICK; SUTCLIFFE; OBSTFELD, 2005), learning and improvisation.

A similar point of view is presented by Eisenhardt and Martin (2000). They identified that management strategies vary with market dynamism. There are organizations – those in moderately dynamic markets – that can rely on dynamic capabilities[3] based on codified and detailed routines. Tacit knowledge may be documented and detailed tasks can be assigned to people. On the other hand, there are organizations that must survive in high-velocity markets – such as software development companies. Their dynamic capabilities cannot rely on existing knowledge, as it changes continuously. Knowledge must be specifically created for each situation (EISENHARDT; MARTIN, 2000). As we presented earlier, in complex systems, this ability for creativity and innovation happens in a state which is called the edge-of-chaos: “a paradoxical space of simultaneous stability and instability” (STACEY, 1996, p. 97).

To accomplish such simultaneous stability and instability, an organization cannot be completely unstructured. Management has to face the challenge of creating “semi-structures” that provide enough structure to help people make sense of the situation, and make them confident to act in highly uncertain situations (EISENHARDT; MARTIN, 2000).

However, this managerial action is not trivial. A great deal of research has been performed to describe and understand how managers can create space for innovation by combining stability and instability in the organizational environment, as shown in the next subsection.

3.4 Organizational ambidexterity

More than two decades ago, March (1991) identified that organizations, as adaptive systems, need to combine stability and instability to survive and prosper. He showed that the development and use of organizational knowledge must be balanced between exploitation and exploration. Exploitation includes activities such as “refinement, choice, production, efficiency, selection, implementation, execution” (MARCH, 1991, p. 71); and on the other hand, exploration is characterized by “search, variation, risk taking, experimentation, play, flexibility, discovery, innovation” (MARCH, 1991, p. 71).

[3] Dynamic capabilities are the strategies that management applies to reassemble resources: physical, human and organizational assets (EISENHARDT; MARTIN, 2000)

The ability to simultaneously pursue exploitation and exploration has currently been named “organizational ambidexterity” (TURNER; SWART; MAYLOR, 2013). Ambidextrous organizations are recognized to be successful because they are aligned and efficient in their management of current demands while simultaneously adaptive to changes in the environment (RAISH; BIRKINSHAW, 2008). This ability provides the organization with the capability to prosper for long periods (TUSHMAN; O’REILLY III, 1996). Organizational ambidexterity is currently taking shape as a research paradigm in organizational theory (RAISH; BIRKINSHAW, 2008) and has already been recognized as an adequate approach to the uncertainty of the software product market (HARRIS; COLLINS; HEVNER, 2009).

Gibson and Birkinshaw (2004) state that one of the ways to achieve ambidextrous behavior is to implement “structural ambidexterity”, in which the organization is structured to have units dedicated to exploitation and separate units dedicated to exploration. Another means is to achieve “contextual ambidexterity”, which is the “behavioral capacity to simultaneously demonstrate alignment and adaptability across an entire business unit” (GIBSON; BIRKINSHAW, 2004, p. 209). In their point of view, alignment creates coherence among all the patterns of activities in the business unit. It means that goals are clear to everyone. Adaptability is the capacity to reconfigure activities in the business. Management systems are, therefore, flexible enough to change work processes in order to meet changing demands (GIBSON; BIRKINSHAW, 2004).

There is a challenge, though, of describing exactly how managers implement ambidexterity. Recent research has shown that organizations have idiosyncratic implementation strategies to be ambidextrous (GIBSON; BIRKINSHAW, 2004; GÜTTEL; KONLECHNER, 2009). Gibson and Birkinshaw (2004), as we have shown, have identified contextually ambidextrous behavior by analyzing the ability of a business unit to be aligned and adaptable at the same time. Tiwana (2008), differently, has identified ambidexterity in information flow among individuals, which can be “strong ties” or “bridging ties”. Bridging ties lead to diversity of accessible knowledge, while strong ties lead to knowledge integration (TIWANA, 2008).

Tiwana has also characterized ambidexterity as a combination of formal and informal controls (TIWANA, 2010). Formal control mechanisms can be accomplished through outcome control, in which there is a specification of the desired result; or through behavioral control, in which there is a specification of the processes or the

steps that should be followed to achieve the expected result. On the other hand, informal controls, named as clan controls, rely on common values, beliefs and shared goals (OUCHI, 1979; TIWANA, 2010). Other complementary views of ambidexterity can also be seen in other studies (see GÜTTEL; KONLECHNER, 2009, DECLERCQ; THONGPAPANL; DIMOV, 2013, BONESSO; GERLI; SCAPOLAN, 2014 and TURNER; MAYLOR; SWART, 2015).

The diversity of ways in which research has been describing the need to combine stability and instability – or alignment and adaptability, or bridging ties and strong ties, or formal controls and clan controls – shows that complex adaptive systems dynamics has been noticed in organizations. There is a need to prevent the system from falling into chaos by using alignment and stability, in other words, by setting clear goals; while also having the courage to allow disorder to generate the emergent novelty, i. e., with adaptability and instability.

In addition, new management tools – as probe/sense/response or sensemaking/learning/improvisation – put into practice strategies that deal with the impossibility of planning work routines in the long term. There is a need to create knowledge for specific situations instead of codifying knowledge that will soon become obsolete.

Agile teams are inside an organizational context where self-organization is valued and change is welcome, as stated by the Agile Manifesto (ref BECK et al). Describing the dynamics these teams present when evolving to maturity might benefit from concepts and theories observed in this chapter, and this is the reason we propose our conceptual framework.

3.5 Conceptual framework

Our conceptual framework for analysis of agile software development maturity is based, hence, on the rationale presented in this chapter. Agile software development teams are complex adaptive systems. They are driven by a self-organized behavior, which challenges management into using strategies that fosters experimenting and learning. The evolution – and the maturing – of this system is a discontinuous process of combining exploitation and exploration. If this combination is successful, it will lead to higher performance through ambidextrous abilities. The following four statements summarize our conceptual framework for data analysis, as shown in Figure 7.

1. An agile software development team is a complex adaptive system that evolves in a discontinuous manner, i.e., there is no such a linear path to evolve;
2. Work processes are improved through experimentation, which is a way to probe new solutions;
3. High performance is achieved through ambidexterity; and,
4. To accomplish that, dynamic capabilities must be developed without codified routines; thus, the focus is on the outcomes pursued by teams.

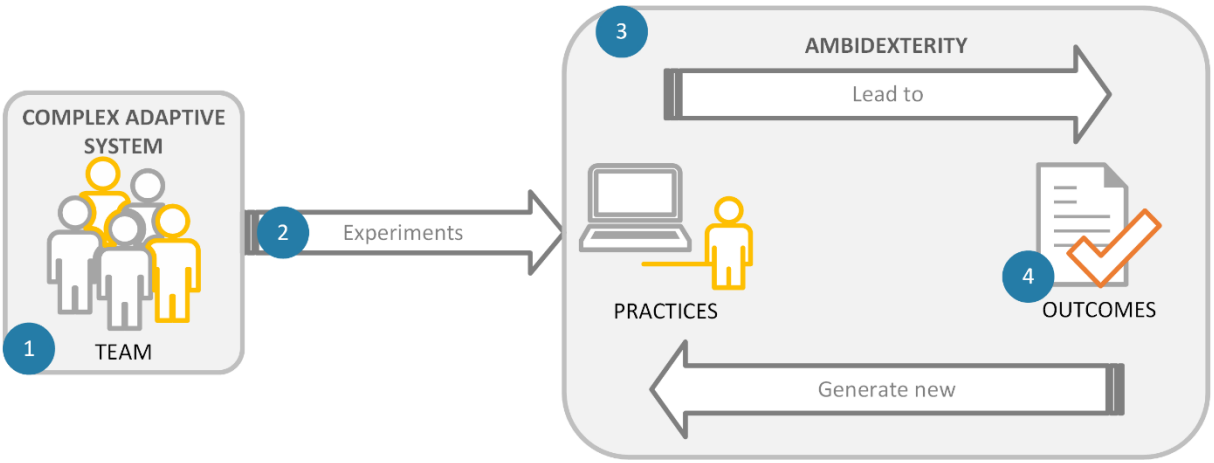


Figure 7 - Conceptual framework

CHAPTER 4. RESEARCH APPROACH

The previous chapter presented the theoretical rationale we used to analyze our empirical data. Before describing our findings in CHAPTER 5, this chapter presents how we organized our research in stages and describes the methods we applied, as shown in Figure 8.

Figure 8 - Contextualization of Chapter 4

This study was guided by the objective to *characterize maturity in agile software development*. To accomplish that, two research questions were used to drive the research work: "*what* is maturity in agile software development?" and "*how* do agile teams get mature?" (Figure 2).

According to Yin (2005), our first question is better answered by methods such as survey and file analysis. The second question is better answered by methods such as experiments, historical research or case studies. We chose the survey method to answer the "what" question, and the case study method to answer the "how" question.

Hence, our overall research strategy was a *mixed methods research approach* (BRYMAN, 2012) because our two research questions have a different nature. The data collected and the methods of analysis were both quantitative and qualitative.

In the field of information systems research our dissertation can be considered as applying the behavioral-science paradigm, as it "seeks to develop […] theories that explain […] organizational and human phenomena surrounding the analysis, design, implementation, management, and use of information systems" (HEVNER et al., 2004, p. 76). We have identified, however, that we have crossed the domain of the design-science paradigm, as the final product of the dissertation is an artifact that might be applied to aid the endeavors surrounding information systems development (HEVNER et al., 2004).

Figure 9 shows our overall research approach. The research was divided into three stages, guided by the research questions, as explained in the next subsections.

4.1 Stage 1 – What is maturity in agile software development?

This was the first stage of the research, in which we were searching for a definition for agile software development maturity, given the variety of concepts we found in the literature (as seen in CHAPTER 2). Results of this stage are published in Fontana et al. (2014b) and presented in CHAPTER 5.

Since it was characterized as an exploratory pursuit of a concept, we used the survey as a research method, according to the guidelines provided by Forza (2002) and Kitchenham et al. (2002). The survey was conducted with agile practitioners, through a questionnaire. This questionnaire had two sections: the first listed eighty-five software development practices (based on a literature review on agile practices) to be evaluated by respondents; and the second section had a single open-ended question (APPENDIX A shows the statements evaluated by practitioners and reference authors).

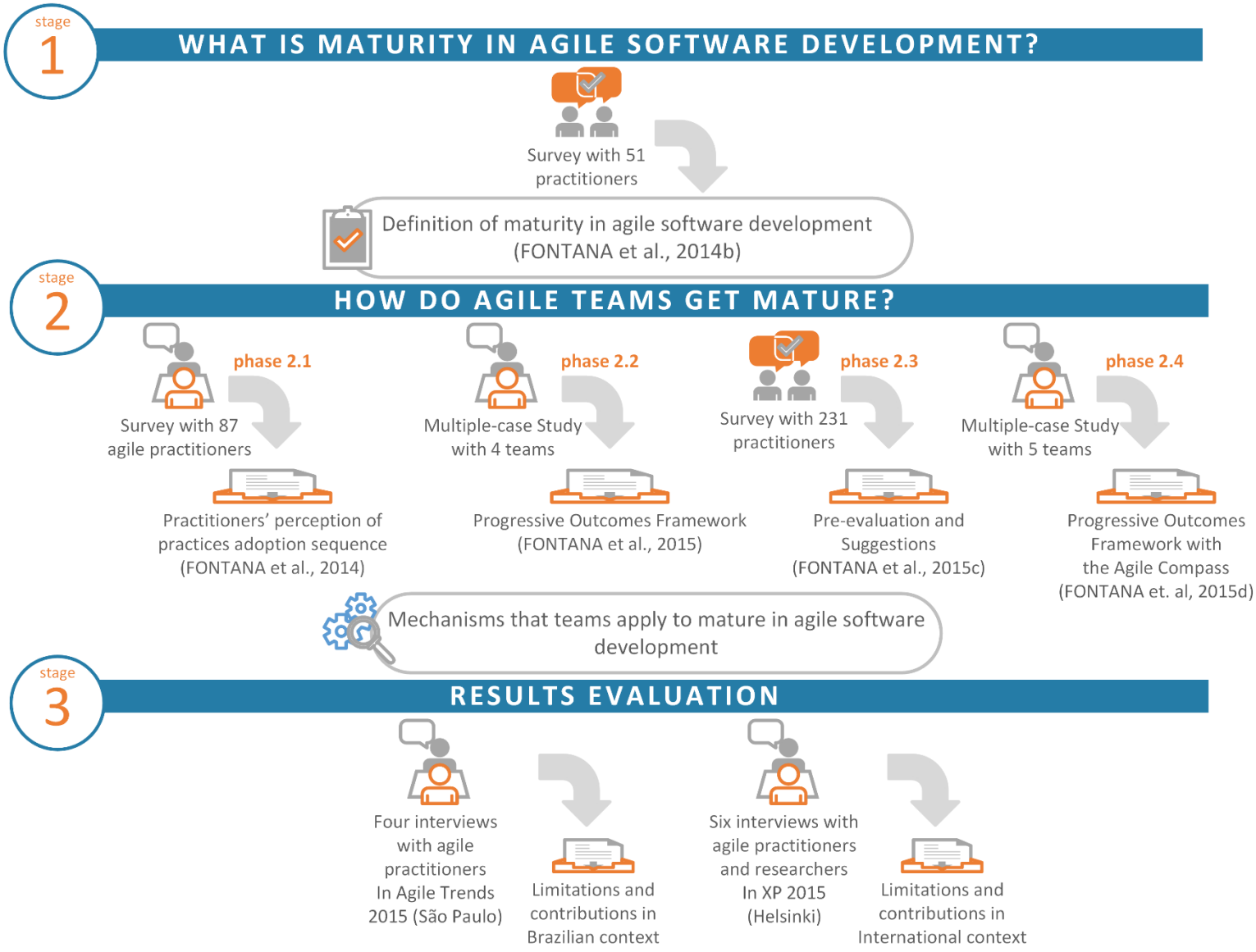


Figure 9 - Research approach

In the first section of the questionnaire, the respondents had to evaluate the perceived maturity of each practice. Then, on a five-point Likert scale, they classified each practice as 1 (“No maturity”), 2 (“Somewhat Mature”), 3 (“Mature”), 4 (“Very Mature”), or 5 (“Very High Maturity”). In the second part of the questionnaire, they answered an open-ended question: “Based on your experience, what is agile software development maturity?”. We conducted a pilot test of the questionnaire with a group of fifteen members of a software engineering research group. They tested both the printed and the on-line formats. Their suggestions helped to improve the layout and the understanding of the practices.

The questionnaires were made available online, using the Qualtrics tool and the snowball technique; the paper format was applied in graduate programs in the Software Engineering field. All respondents were required to have previous experience in agile software development.

As we had two types of data: quantitative – in the classification of the practices; and qualitative – in the definition each respondent gave to the open-ended question, we conducted two distinct data analysis:

- The quantitative data analysis referred to the need of clustering the practices in the questionnaire, considering the maturity classifications they received. To accomplish that, statistical cluster analysis was applied. This technique groups data according to the similarities among them (HAIR et al., 2006; JOHNSON; WICHERN, 2007). In our analysis, Ward’s clustering method was used and the clustering variable, used to evaluate similarity, was the classification of maturity given by respondents. Similarity was measured using squared Euclidian distance. We achieved a twenty-cluster solution, which was validated using the Davies-Bouding index (DAVIES; BOULDIN, 1979) and a sanity check that verified Cronbach’s Alpha within clusters (BLAND; ALTMAN, 1997). After the clusters had been defined, we calculated how they were related to maturity. For each practice, we calculated the percentage of responses corresponding to classifications 1 or 2, 3, and 4 or 5. Based on the percentages of the practices, we calculated the mean percentages of the cluster;

- The qualitative data analysis was focused on the need to verify the concepts that respondents used to define maturity in agile software development. To accomplish that, we applied the content analysis technique (BARDIN, 2011). We codified the answers and identified the frequency of occurrence of each code using the software *NVivo*. A number of categories emerged from this analysis. Then, we also calculated the frequency of occurrence of each category. In the last step of the content analysis, we identified how the main codes related to each other.

Thus, by using the clusters that were classified with the highest maturity levels and the concepts that practitioners most often cited in their responses, we proposed a definition for maturity in agile software development (see Stage 1 in Figure 9).

4.2 Stage 2 – How do agile teams get mature?

Stage 2 was focused on answering our second research question, which inquires how agile teams get mature. This stage comprised four phases: Phase 2.1, in which we had a preliminary insight about the roadmap to agile maturity; Phase 2.2, in which we performed the first round of case studies; Phase 2.3, when we pre-evaluated our findings with agile practitioners; and Phase 2.4, in which the second round of case studies was performed (see Figure 9).

4.2.1 Phase 2.1

The objective of this phase was to gain insight about how agile software development practitioners see the roadmap to maturity. We conducted a survey at an agile event – the Agile Trends 2013 – in São Paulo, Brazil. The data were collected using printed questionnaires (APPENDIX B). This questionnaire collected the following information:

- personal information;
- practitioners' opinion about the usefulness of a maturity model for agile software development; and,
- a list of practices, to which the respondents assigned a sequence. This sequence meant the suggested roadmap that teams should follow to mature in agile software development.

To build the list of practices, we reviewed the literature on how agile practices have been evaluated and measured (LAYMAN; WILLIAMS; CUNNINGHAM, 2004; WILLIAMS et al., 2004; WILLIAMS; RUBIN; COHN, 2010; ABBAS; GRAVEL; WILLS, 2010; BUGLIONE, 2011; SOUNDARAJAN; ARTHUR; BALCI, 2012; KETTUNEN, 2012). We consolidated these practices using a mind mapping tool and grouped them according to the Software Engineering Body of Knowledge (SWEBOK) areas (BOURQUE; FAIRLEY, 2014). The different practices used by authors in the literature were, then, translated to empirical domain and grouped in issues where an agile team could emphasize work to develop maturity. The resulting group was a list of 13 practices (see question 2 in APPENDIX B). Respondents had, hence, to number the practices in the ideal sequence of adoption, based on their experience. They could leave practices in blank (meaning they are not relevant) or add other practices.

Three hypothesis tests – using the Chi-Square test – were applied to verify the following null hypothesis:

- Hypothesis 1 – The practice is as likely to be considered relevant (to be numbered) as not relevant (to be left blank);
- Hypothesis 2 – All numberings have the same probability to be chosen by respondents (there is no preference for a particular classification);
    - For this hypothesis test, when we identified there was a numbering trend, we grouped the numbering gave by respondents as “essential” (when numbered as 1, 2, or 3); “intermediate” (when numbered as 4, 5, 6, 7, 8, or 9); and “desirable” (when numbered as 10, 11, 12 or 13). For each classification (essential, intermediate or desirable), the average percentage of responses given to inner practices was calculated;
- Hypothesis 3 – There is the same distribution of numberings for experienced practitioners and for non-experienced practitioners.

With these tests, we could: i) verify whether the practices were relevant to maturity in agile software development; ii) identify a trend in the numbering of the practices and place them in an essential, intermediate or desirable level of agile maturity; and iii) verify whether the opinions of inexperienced practitioners are different from those of experienced practitioners. In addition to that, we performed an analysis on the usefulness of an agile software development maturity model with an open-ended question answered by practitioners.

The result of this study was practitioners' perception of the sequence of adoption of practices as well as some insight about the practical relevance of the theme of this dissertation. The results were published in Fontana et al. (2014).

4.2.2 Phase 2.2

Based on the preliminary insights of Phase 2.1 and on the Conceptual Framework, we conducted four case studies in this phase. The objective was to answer the research question on how agile teams get mature, but based on the observation of real cases this time, rather than on practitioners' opinions. The findings of this phase were published in Fontana et al. (2015).

We chose the case study method because it is the appropriate approach to answer "how" questions (YIN, 2005) and to understand the dynamics present within single settings (EISENHARDT, 1989). As we were seeking to replicate patterns across different settings, we chose to study multiple cases.

The unit of analysis was the agile software development team. We chose companies or teams with different profiles because we wished to find patterns of agile maturing across different contexts. The definition of these contexts is based on the classification of software services in the Brazilian industry (SOFTEX, 2012). From all the possible activities, the ones that regard software development are 1) development of software on demand; 2) development of customizable software; 3) development of non-customizable software; and 4) web portals and internet information services. In this first round of studies, we investigated the first three contexts. The company that develops web portals was included in the second round, described in Section 4.2.4.

Hence, the teams could have any size, to apply any agile method, with any length of experience in agile adoption. Our decision on not to choose an specific agile method is based on evidence that agile methods are currently being highly customized, and further scientific research should not focus on specific methods (KURAPATI; MANYAM; PETERSEN, 2012; BUSTARD; WILKIE; GREER, 2013). The teams were chosen, then, based on the classification of the service their companies provide, on the acknowledgement of their companies' practices in agile community and on the ease of access to the field. The openness of the companies to the research procedures was also a choice criterion, as team members would be more prone to collaboration. Figure 10 summarizes our choice criteria.

Figure 10 - Choice criteria

The propositions we defined *a priori* to analyze the cases (YIN, 2005) were based on the Conceptual Framework shown in Figure 7 and on the results obtained in Phase 2.1 (FONTANA; REINEHR; MALUCELLI, 2014), as shown in Figure 11. This figure shows the four propositions, the conceptual framework point each proposition relates to (see Figure 7 on page 46), and the authors that support them.

Proposition 1 (Based on 1 in Figure 7)

The team plays a central role in agile software development maturity

MCHUGH, CONBOY, LANG, 2012; MIDDLETON; JOYCE, 2012; PACKLICK, 2007; SUOMINEN; MÄKINEN, 2013

Proposition 2 (Based on 2 and 3 in Figure 7)

Teams get mature in agile software development by combining exploration and exploitation activities, that is, through ambidexterity

MARCH, 1991; TURNER; SWART; MAYLOR, 2013; GIBSON; BIRKINSHAW, 2004; TIWANA, 2008; VIDGEN; WANG, 2009

Proposition 3 (Based on 4 in Figure 7)

The exact set of practices is not predefined at each maturing stage

EISENHARDT; MARTIN, 2000; COLEMAN; O'CONNOR, 2008; KIRK; TEMPERO, 2012; BUSTARD; WILKIE; GREER, 2013; FONTANA; REINEHR; MALUCELLI, 2014

Proposition 4 (Based on FONTANA; REINEHR; MALUCELLI, 2014)

Teams evolve in agile development by departing from agile values, involved customer, planning and requirements; and then, they later invest in agile coding and agile testing

Figure 11 - The propositions of the multiple-case study

The approach for the case studies was based on the stages suggested by Eisenhardt (1989), and it is summarized in Figure 12.

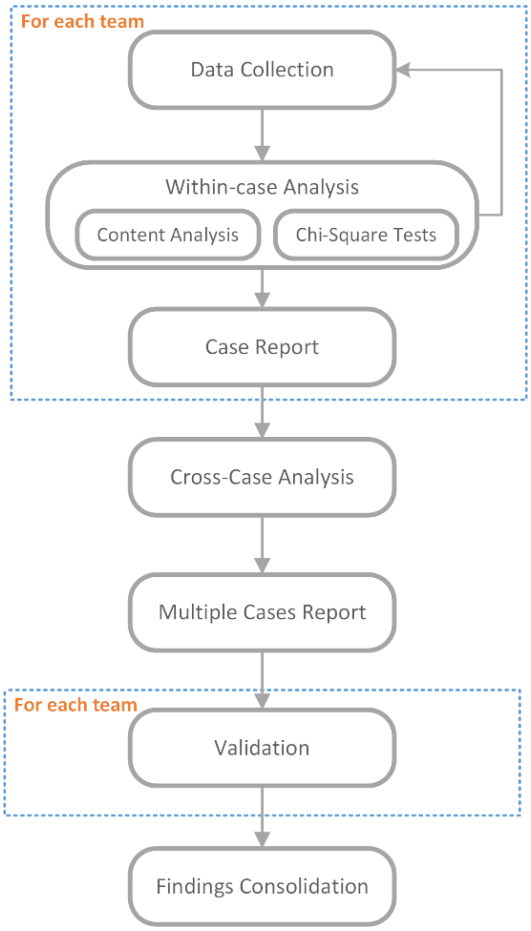


Figure 12 - Research approach for the case studies

Data Collection: We collected qualitative and quantitative data from each team. The qualitative approach comprised interviews with three team members: the team leader, the most experienced developer and the least experienced developer. The individuals who participated were indicated by the team leader. The interviews were semi-structured: we asked the interviewee to tell us a story of how the adoption of the practices evolved since the team started using agile methods. As a guide to data collection, we used the framework suggested by Kirk and Tempero (2012), in which the practices are classified according to their function: to define the software, to make the software or to deliver the software. We also kept track of the practices applied in the past, in the present and in the future. The quantitative approach comprised two types of questionnaires applied to the whole team. The team leader provided personal

information, company information, ambidexterity perception and project success perception. The other team members provided personal information and ambidexterity perception. The questions regarding ambidexterity information were based on ambidexterity studies (see APPENDIX C) and the ones regarding project success information were based on the success factors for agile methods identified by Misra et al. (2009). They were both answered on a five-point Likert scale: from 1 (completely disagree) to 5 (completely agree).

Within-case Analysis: Our qualitative data were analyzed using the content analysis technique, and the quantitative data were analyzed using Chi-Square tests. For the qualitative data analysis, all the interviews were recorded and transcribed. They were stored on Atlas TI, which was the tool we used to perform content analysis (BARDIN, 2011). Our content analysis comprised: i) scanning the text; ii) codifying; iii) assigning each code to a category: define, make or deliver (KIRK; TEMPERO, 2012); iv) assigning each code to either past, present or future; v) creating memos when necessary; and, vi) making associations with other related codes. Three networks of codes were created for each team: past, present and future (see an example in APPENDIX D). The quantitative analysis consisted in applying Chi-Square tests for each of the aspects evaluated in the questionnaire: performance, alignment, adaptability, bridging ties and strong ties. The null hypotheses were:

- Null Hypothesis 1 - This team disagrees on performance results (same probability of agreement, disagreement or neutral opinion regarding the evaluation of statements);
- Null Hypothesis 2 - This team disagrees on good alignment perception;
- Null Hypothesis 3 - This team disagrees on bad alignment perception;
- Null Hypothesis 4 - This team disagrees on adaptability perception;
- Null Hypothesis 5 - This team disagrees on the perception of bridging ties;
- Null Hypothesis 6 - This team disagrees on the perception of strong ties.

All the tests that resulted in a p-value below 0.05 were rejected. For us, it meant that the team had a consolidated perception of the evaluated aspect. Thus, we expected all of the null hypotheses to be rejected. That is, the whole team should agree on the perception of performance, alignment, flexibility, strong ties and bridging ties. In addition to that, the highest percentage of respondents should agree positively on each of the evaluated aspects. The quantitative analysis was included in the Case Report in

the form of graphs that were presented to the team leader (see an example in APPENDIX E).

Cross-case Analysis: As suggested by Eisenhardt (1989), in the cross-case analysis we sought for replication across cases. To accomplish that, we used a mind mapping tool. The outcomes identified in each within-case analysis were inserted on the map and evidence to the original team was appropriately recorded. The outcomes that appeared in the past were listed before the ones that appeared in the present. The ones in the present were mapped before the ones in the future. As the map was being built, the replication of outcomes across cases was identified and the following types of outcomes emerged: practices, team, deliveries, requirements, product and customer[4]. APPENDIX F explains the procedure applied in the cross-case analysis. The multiple-cases report was created in a consolidated mind map, partially presented in APPENDIX G.

Validation: As the outcomes identified in the content analysis were inferred based on the stories told by practitioners and the practices we codified, it was necessary to validate these outcomes. Thus, we conducted one more interview with each team leader to show the Case Report and receive feedback. In general, all findings were positively validated with few changes. The transcriptions of the validation interviews were also stored on Atlas TI, codified and, whenever necessary, code networks and individual cases reports were updated.

The main result of Phase 2.2 was the Progressive Outcomes framework for maturing in agile software development, published in Fontana et al. (2015).

4.2.3 Phase 2.3

The pre-evaluation of the results from the first four cases were performed through surveys in agile events in Brazil. We conducted three talks in which we presented:

- Our definition of maturity in agile software development;
- The concept of ambidexterity for managing agile teams; and
- The Progressive Outcomes framework;

[4] The naming of the outcomes categories was later reviewed, based on the results evaluation performed in Stage 3.

After the talks, we collected the feedback from practitioners through a questionnaire with two parts. The first part collected personal data: name, e-mail, company, city, position, length of experience in software engineering and length of experience in agile methods. The second part presented seven statements that should be evaluated on a five-point Likert scale, ranging from completely disagree to completely agree. The statements were based on design-science validation guidelines (HEVNER et al., 2004), which pose that utility, quality and efficacy of the artifact should be evaluated. Table 3 shows the statements evaluated by respondents for these aspects. In the third event, we included and open-ended question for respondents to make any comments, express their opinions, or give suggestions.

Table 3 - Statements evaluated in pre-evaluation survey

Aspect to be evaluated	The Progressive Outcomes framework…
Utility	… is useful to aid teams to evolve with agile methods
	… is useful to define what maturing is in agile
Quality	… is easy to understand
	… comprises what I believe is necessary to evolve in agile
	… includes unnecessary information
Efficacy	… allows me to identify the current situation on my team/in my company
	… is adaptable to different organizational contexts

The events where we collected data were Agile Tour in Curitiba (September, 2014), Agile Tour in Campinas (October, 2014) and Agile Brazil in Florianópolis (November, 2014).

The collected data were analyzed using descriptive statistics. We identified the percentage of respondents who agreed on each statement (adding the amount of answers as “completely agree” to the amount of “partially agree”), the percentage of respondents who disagreed on each statement (adding the amount of answers as “completely disagree” to the amount of “partially disagree”), and kept the percentage of respondents with “no opinion”.

We published the results of this pre-evaluation in Fontana et al. (2015c).

4.2.4 Phase 2.4

In this phase, we chose additional cases, following the same criteria described in Figure 10: classification of the service companies provide, the acknowledgement of companies’ practices in agile community and ease of access to the field. We were invited by one of the companies to conduct the research after our talk in Agile Brazil. Thus, we added four cases which we chose to the study, plus one by invitation.

These five additional studies followed the same protocol described in Phase 2.2, with one exception. We replaced the interview with the least experienced developer with an interview with another experienced developer, or another experienced role in the team. This is due to the fact that the interviews with inexperienced people generated few codes in the content analysis and, thus, gave little contribution in the identification of the evolvement of practices. With this change, results were improved with interviews that generated more codes in content analysis and, thus, more evidence.

Based on the feedback received in the pre-evaluation (Phase 2.3), we included an evaluation of organizational support evolvement for the big companies in our sample (Company C and Company G). For this reason, we performed a complementary interview in Company C and asked Company G additional questions concerning organizational support in the past, in the present and perceptions for the future. We included, as a result, the *organizational support* category in the Progressive Outcomes framework.

Also based on the feedback received in the pre-evaluation (Phase 2.3), we summarized, for each outcome, how a team could identify whether it was accomplished or not. The result was a tool for identifying maturity in agile software development, which we called “Agile Compass”. The procedure to create this tool was the following:

1. Using a mind mapping tool, we grouped all evidences that led us to conceive each outcome. To do that, all qualitative data from the within-case analysis of the case studies was consolidated in a map. APPENDIX I shows an example: the map with evidences we grouped for the outcomes in “Pace of deliveries” category;
2. We summarized the group of evidences in a sentence that gives a definition for the outcome, and two or three statements for the team to check whether the outcome was accomplished or not;

The Agile Compass is presented in Section 5.3 and was published in Fontana et. al. (2015d).

4.3 Stage 3 – Results Evaluation

This research sought to investigate what agile maturity is and how agile teams evolve. Our final product is a framework that considers the findings for each of these questions. It is a designed artifact (HEVNER et al., 2004) and, for this reason, we

applied the suggestions given by Venable et al. (2012) for evaluation in design-science research.

As a first step for a designed artifact evaluation, Venable et al. (*Ibid*.) state that the researcher should characterize the context of the evaluation. In this respect, the evaluation of our results is characterized by:

- Artifact: the description of the mechanisms for maturing in agile software development and the underlying maturity concept;
- Purpose: to "evaluate the formalized knowledge about the utility of a designed artifact for achieving its purpose" (VENABLE; PRIES-HEJE; BASKERVILLE, 2012, p. 425);
- Ex-ante evaluation: it regards the evaluation of an artifact which was not instantiated, given the infeasibility of testing the maturity framework in real settings within the dissertation development period;
- Naturalist evaluation: we involved practitioners in the evaluation and, thus, explored the perceived performance of a situation within the organization;
- Method: the evaluation was performed through interviews with agile practitioners and researchers.

Semi-structured interviews with agile practitioners and researchers were used to investigate the utility, quality and efficacy of the artifact, based on design-science validation guidelines by Hevner et al. (2004). We also included an evaluation of the applicability of the findings in international context, given the limitation of the sample with Brazilian teams. For this reason, we conducted ten interviews:

- Four interviews with Brazilian agile consultants/practitioners, conducted during the event Agile Trends 2015, in São Paulo, Brazil;
- Six interviews with agile researchers and practitioners, from Brazil and other countries, conducted during the XP 2015 conference, in Helsinki, Finland;

We presented our research method and findings (using printed slides in A3 format) and, later, argued interviewees about results utility, quality and efficacy. Table 4 shows the questions we used as a guide for the interviews. This table also presents the questions we asked to identify locality idiosyncrasies in our results.

Table 4 - Questions for evaluating the research findings

Aspect to be evaluated	Questions about the aspect
Utility	In which contexts do you think this framework is useful?
	How would you use this framework?
	Do you think a certification of agile teams would be applicable?
Quality	What did you not understand in the framework?
	What kind of information do you think was lacking in the framework?
	Would you add any content to the framework?
	Which information do you think the framework should not provide?
Efficacy	How do you perceive maturity concept in this framework?
	How do you think a team could implement the framework to get mature?
	How do you perceive the flexibility of the framework to different organizational contexts?
Locality	Which are the differences in locality that you perceive, considering your country/region?
	How do you think this framework is applicable to your country/region?

We identify, therefore, that our research was a behavioral-science study, with a characteristic of design-science research, as a designed artifact was created. Our mixed-methods approach comprised three stages, namely: Stage 1, whose objective was to answer our first question – *what* is agile maturity; Stage 2, which aimed at answering the second research question – *how* is agile maturity gained; and a third stage to evaluate the findings. The next chapter presents our findings.

CHAPTER 5. RESEARCH RESULTS

After the research approach has been presented, this chapter describes our findings, as summarized in Figure 13. It first presents the data that support the answer for the first research question and, later, the data that support the findings for the second research question. The chapter ends by reporting how agile practitioners and researchers evaluated the findings.

Figure 13 - Contextualization of Chapter 5

5.1 What is maturity in agile software development?

The analysis of the data we collected at the exploratory stage of this dissertation has shown that:

Maturity in agile software development means having an experienced team that:

- *collaborates on projects by communicating and being committed;*
- *cares about customers and software quality;*
- *allows requirements to change;*

- *shares knowledge;*
- *manages source code and tests using tools, methods and metrics supported by infrastructure that is appropriate for agility;*
- *self-organizes at a sustainable pace;*
- *standardizes and continuously improves agile practices; and,*
- *generates perceived outcomes for customers and management.*

To achieve this definition (FONTANA et al., 2014b), we combined qualitative and quantitative data analyses.

5.1.1 Quantitative data analysis

As explained in Section 4.1, we collected data through a questionnaire and received fifty-one responses. Our sample represented thirty-three different Brazilian companies and four multinational companies that develop software primarily for their own use. Most of the respondents were developers and had up to three years of experience in agile methods, mainly Scrum. Table 5 shows the respondents' profiles.

Table 5 - Profile of respondents to exploratory survey (FONTANA et al., 2014b)

		Number of respondents	Percentage
Role in their team	Developer	18	35%
	System Analyst	12	23%
	Leader	8	16%
	Software Architect	5	10%
	Test Analyst	4	8%
	Others	4	8%
Experience in agile methods	From 1 to 3 years	27	53%
	Less than 1 year	9	17%
	From 4 to 6 years	8	16%
	More than 6 years	7	14%
Agile method	Scrum	36	70%
	XP (Extreme Programming)	8	16%
	Others (Kanban, customized methods)	7	14%
Experience in SPI Models*	CMMI-DEV	24	47%
	MPS.BR	11	22%

*Multiple responses allowed

We used the cluster analysis technique to group all practices, considering the maturity classification to which they were assigned. As a result, we obtained twenty clusters of practices and named them according to the characteristic of the inner practices. The clusters are shown in Table 6.

Table 6 also shows the maturity classifications each practice and each cluster received. For example, the practice "X41 – Communicating face-to-face daily" received 19.6% of the classifications as "No Maturity" or "Somewhat mature", 35.3% of the classifications as "Mature" and 45.1% as "Very Mature" or "Very High Maturity". In the table, they are shown respectively in the columns "Low", "Medium" and "High".

The cluster means were calculated based on the percentages of cluster's inner practices. Table 6 shows highest to lowest maturity clusters.

The lowest maturity cluster is "Traditional Analysis", as it received 56.9% of classifications as low-maturity. It contains just one practice that represents traditional systems analysis. The next cluster is "Lightweight Requirements", which contains practices relative to using metaphors and lightweight requirements. It was classified as low-maturity by 55.9% of respondents.

The next cluster is "Caring about the Code". It received a classification of medium-maturity by 45.6% of respondents. It includes practices relative to the quality of the code, such as technical design, inspections, reviews and tests, as well as appropriate distribution of expertise on the team. The "Customer Presence" cluster groups practices relative to having the customer actively participate in the projects and physically close to the team, and was classified by 47.1% of respondents as a low-maturity cluster.

"Agile Coding" and "Physical Distribution" clusters had similar classifications of low-, medium- and high-maturity, around 30% each. The former cluster groups agile coding practices, such as refactoring, pair programming, focused work, and agile quality assurance. The latter refers to practices that describe the physical distribution of the team, either collocated or distributed.

The next cluster received a medium-maturity classification by 37.3% of respondents. We named it "Project Monitoring". It contains practices relative to tracking project progress, having a defined process to monitor the project, and using metrics to ground decisions. Next, Table 6 shows the "Agile Project Management" cluster, which was classified by 35.9% of respondents as low-maturity. They refer to agile estimation and agile planning practices.

Growing in perceived maturity, "Traditional Software Process" is the next cluster, with 43.1% of classifications as medium-maturity. This cluster mostly groups practices derived from CMMI-DEV process areas and the ones relative to dealing with complexities in the organization. Examples of practices considered as medium-maturity by respondents include having defined process assets, establishing and maintaining plans, evaluating processes and product objectively, managing risks and managing alignment between requirements and the work products.

The next cluster is the first that was classified by the majority of respondents as high-maturity: "Manage Requirements". It received 44.9% of high-maturity

classifications and includes practices relative to defining requirements in agile ways (such as product backlog and user stories), but also practices relative to planning with timeboxes, being multidisciplinary and developing peoples' skills.

Also classified as high-maturity practices, "Iterations" and "Meetings" clusters are shown in Table 6. They received 45.8% and 46.1% of high-maturity classifications, respectively. The iterations practices regard having short software releases in an incremental development; and meetings regard holding agile-like daily meetings and retrospective meetings.

The "Simplicity" cluster comes next, with 47.1% of classifications as high-maturity, grouping practices such as using simple software design and having communication-based work. The practices relative to "Performance Analysis" have also been classified as high-maturity by 48% of respondents.

The next six clusters were classified by the majority of respondents (more than 50%) as high-maturity practices in agile software development. "Sustainable Self-organization" received 50.8% of classifications as high-maturity. The practices of this cluster comprise a number of ways for agile teams to behave, such as collective code ownership, maintaining a sustainable pace, self-organizing, giving continuous feedback etc. "Test-driven Development" and "Caring about the solution" had around 51% of classifications as high-maturity. They are small clusters that group practices relative to test automation and refers to the quality of the solution, respectively.

The top-three high-maturity practices are represented by the clusters "Management of Code and Tests", "Emerging Requirements" and "Collaboration". The first cluster received 52% of classifications as high-maturity. It comprises some planning practices, running user acceptance tests, collecting tests metrics, and managing source code. The "Emerging Requirements" cluster represents allowance of requirements to evolve and emerge, with 52.9% of classification as high-maturity.

Overall, the highest maturity classification goes to the "Collaboration" cluster. Respondents consider practices such as communicating face-to-face, collaborating, keeping work simple, sustaining autonomy and sharing responsibility as having the highest maturity. These practices received an average of 56.9% of classifications as high-maturity.

Table 6 - Clusters and maturity assignments (FONTANA et al., 2014b)

Cluster Name	Practice Number	Practice Description	Percentage (practice)[a]			Total	Mean percentage (cluster)[b]		
			Low	Medium	High		Low	Medium	High
Collaboration	X41	Communicating face-to-face daily	19.6%	35.3%	**45.1%**	100%			
	X42	Questioning and learning from one another	7.8%	29.4%	**62.7%**	100%			
	X43	Collaborating with team members	7.8%	25.5%	**66.7%**	100%			
	X44	Keeping work simple	9.8%	35.3%	**54.9%**	100%			
	X45	Not losing autonomy when under pressure to meet deadlines	19.6%	29.4%	**51.0%**	100%			
	X46	Sharing responsibility	17.0%	23.5%	**58.8%**	100%			
	X62	Encouraging a culture of working together as a team rather than individually	9.8%	31.4%	**58.8%**	100%	13.2%	30.0%	**56.9%**
Emerging Requirements	X2	Allowing requirements to evolve during the project	13.7%	35.3%	**51.0%**	100%			
	X5	Allowing the emergence of requirements	17.6%	27.5%	**54.9%**	100%	15.7%	31.4%	**52.9%**
Management of Code and Tests	X19	Running user acceptance tests	5.9%	35.3%	**58.8%**	100%			
	X21	Collecting test metrics	23.5%	31.4%	**45.1%**	100%			
	X22	Managing software configuration (version control)	11.8%	33.3%	**54.9%**	100%			
	X23	Managing source code	21.6%	31.4%	**47.1%**	100%			
	X25	Planning releases	15.7%	33.3%	**51.0%**	100%			
	X26	Planning before and during the project	15.7%	29.4%	**54.9%**	100%	15.7%	32.4%	**52.0%**
Caring about the Solution	X15	Using code standards	13.7%	33.3%	**52.9%**	100%			
	X16	Being concerned about database architecture	11.8%	39.2%	**49.0%**	100%			
	X32	Defining scope according to schedule	23.5%	23.5%	**52.9%**	100%	16.3%	32.0%	**51.6%**
Test-driven Development	X18	Running automated unit tests	21.6%	23.5%	**54.9%**	100%			
	X20	Doing test-driven development	29.4%	23.5%	**47.1%**	100%	25.5%	23.5%	**51.0%**
Sustainable Self-organization	X12	Using collective code ownership	25.5%	31.4%	**43.1%**	100%			
	X17	Doing continuous code integration	15.7%	25.5%	**58.8%**	100%			
	X34	Maintaining a sustainable pace (do minimum overtime)	15.7%	25.5%	**58.8%**	100%			
	X39	Responding to pressure by re-prioritizing or re-scoping rather than working overtime or adding people	15.7%	31.4%	**52.9%**	100%			
	X48	Self-organizing	21.6%	33.3%	**45.1%**	100%			
	X49	Giving continuous feedback	17.6%	35.3%	**47.1%**	100%			
	X55	Implementing development infrastructure that supports agility	25.5%	27.5%	**47.1%**	100%			
	X56	Developing people's agility skills	21.6%	35.3%	**43.1%**	100%			
	X66	Having the customer actively participate during the project	11.8%	25.5%	**62.7%**	100%			
	X67	Developing products that respond to business needs	9.8%	41.2%	**49.0%**	100%	18.0%	31.2%	**50.8%**
Performance Analysis	X75	Getting to know strengths and weaknesses and be able to plan and implement process improvements based on that	17.6%	35.3%	**47.1%**	100%			
	X76	Identifying gaps in performance and selecting and deploying improvements to close these gaps	15.7%	35.3%	**49.0%**	100%	16.7%	35.3%	**48.0%**
Simplicity	X6	Eliciting requirements based on communication	11.8%	27.5%	**60.8%**	100%			

Cluster Name	Practice Number	Practice Description	Percentage (practice)[a]			Total	Mean percentage (cluster)[b]		
			Low	Medium	High		Low	Medium	High
	X10	Using simple software design	19.6%	**47.1%**	33.3%	100%	15.7%	37.3%	**47.1%**
Meetings	X35	Holding daily progress tracking meetings	29.4%	23.5%	**47.1%**	100%			
	X36	Holding retrospective meetings	23.5%	31.4%	**45.1%**	100%	26.5%	27.5%	**46.1%**
Iterations	X28	Doing iterative and incremental development	15.7%	**43.1%**	41.2%	100%			
	X29	Making short software releases	27.5%	31.4%	**41.2%**	100%			
	X30	Delivering working software continuously	17.6%	27.5%	**54.9%**	100%	20.3%	34.0%	**45.8%**
Manage Requirements	X1	Using product backlog to define requirements	21.6%	37.3%	**41.2%**	100%			
	X3	Using stories to define requirements	27.5%	**39.2%**	33.3%	100%			
	X11	Specifying software architecture	19.6%	35.3%	**45.1%**	100%			
	X27	Using timeboxes in planning	25.5%	33.3%	**41.2%**	100%			
	X33	Making estimates with the people who will do the work	17.6%	33.3%	**49.0%**	100%			
	X61	Being multidisciplinary	13.7%	**43.1%**	**43.1%**	100%			
	X70	Identifying causes of problems and taking actions to prevent them in the future	13.7%	35.3%	**51.0%**	100%			
	X78	Developing people's skills and knowledge so they can perform their roles effectively and efficiently	17.6%	27.5%	**54.9%**	100%	19.6%	35.5%	**44.9%**
Traditional Software Process	X58	Dealing easily with organizational complexity	19.6%	**41.2%**	39.2%	100%			
	X59	Dealing easily with regulatory compliance	15.7%	**45.1%**	39.2%	100%			
	X63	Dealing easily with domain complexity	13.7%	**47.1%**	39.2%	100%			
	X64	Dealing easily with technical complexity	13.7%	**43.1%**	**43.1%**	100%			
	X65	Dealing easily with enterprise discipline	9.8%	**45.1%**	**45.1%**	100%			
	X74	Having defined process assets, work environment standards, and rules and guidelines	25.5%	29.4%	**45.1%**	100%			
	X79	Establishing and maintaining plans that define project activities	23.5%	37.3%	**39.2%**	100%			
	X80	Evaluating processes and work products objectively and addressing non-compliance issues	21.6%	**41.2%**	37.3%	100%			
	X82	Formally eliciting, analyzing, and validating requirements for the product and stakeholders	21.6%	**45.1%**	33.3%	100%			
	X83	Planning and invoking risk handling activities as needed across the life of the project	23.5%	**45.1%**	31.4%	100%			
	X84	Managing the acquisition of products and services from suppliers	25.5%	**47.1%**	27.5%	100%			
	X85	Maintaining alignment between requirements and the plans and work products of the project	17.6%	**51.0%**	31.4%	100%	19.3%	**43.1%**	37.6%
Agile Project Management	X24	Using the planning game	**41.2%**	25.5%	33.3%	100%			
	X31	Making agile project estimates	27.5%	**39.2%**	33.3%	100%			
	X50	Writing agile documentation	**39.2%**	23.5%	37.3%	100%	**35.9%**	29.4%	34.6%
Project Monitoring	X37	Tracking and reporting iteration progress	25.5%	**43.1%**	31.4%	100%			
	X38	Integrating management activities directly into development tasks	23.5%	**45.1%**	31.4%	100%			
	X71	Analyzing possible decisions using a formal evaluation process that evaluates identified alternatives against established criteria	29.4%	**35.3%**	**35.3%**	100%			
	X72	Managing projects according to an integrated and defined process	27.5%	29.4%	**43.1%**	100%			
	X73	Collecting metrics that are used to support management information needs	27.5%	**41.2%**	31.4%	100%			

Cluster Name	Practice Number	Practice Description	Percentage (practice)[a]			Total	Mean percentage (cluster)[b]		
			Low	Medium	High		Low	Medium	High
	X77	Having a quantitative understanding (metrics-based) of processes	27.5%	**37.3%**	35.3%	100%			
	X81	Managing projects with measures and analytic techniques	**41.2%**	29.4%	29.4%	100%	28.9%	**37.3%**	33.9%
Physical Distribution	X57	Being geographically distributed (different cities or countries)	**39.2%**	27.5%	33.3%	100%			
	X60	Distributing physically so as to reflect agile philosophy	29.4%	**37.3%**	33.3%	100%	**34.3%**	32.4%	33.3%
Agile Coding	X13	Doing code refactoring	23.5%	27.5%	**49.0%**	100%			
	X14	Doing pair programming	**35.3%**	33.3%	31.4%	100%			
	X40	Focusing on work (priorities do not change during iteration)	29.4%	**45.1%**	25.5%	100%			
	X47	Doing things when they have to be done, not before	35.3%	**37.3%**	27.5%	100%			
	X54	Doing agile quality assurance	**37.3%**	31.4%	31.4%	100%	32.2%	**34.9%**	32.9%
Customer Presence	X68	Allowing customer to drive iterations	**43.1%**	25.5%	31.4%	100%			
	X69	Customer being collocated	**51.0%**	31.4%	17.6%	100%	**47.1%**	28.4%	24.5%
Caring about the Code	X4	Doing technical design of requirements	31.4%	**49.0%**	19.6%	100%			
	X51	Distributing expertise on the team appropriately	17.6%	**47.1%**	35.3%	100%			
	X52	Running lightweight tests and reviews	39.2%	**52.9%**	7.8%	100%			
	X53	Analyzing and inspecting code	**35.3%**	33.3%	31.4%	100%	30.9%	**45.6%**	23.5%
Lightweight Requirements	X7	Defining lightweight requirements	**43.1%**	35.3%	21.6%	100%			
	X9	Using metaphors to describe requirements	**68.6%**	25.5%	5.9%	100%	**55.9%**	30.4%	13.7%
Traditional Analysis	X8	Performing traditional systems analysis	**56.9%**	29.4%	13.7%	100%	**56.9%**	29.4%	13.7%

[a]This column is divided into three subcolumns: "Low" shows the percentage of responses that associated this practice with level 1 ("No maturity") or 2 ("Somewhat Mature"); "Medium" shows the percentage of responses that associated this practice with level 3 ("Mature"); and "High", the percentage that associated the practice with level 4 ("Very Mature") or 5 ("Very High Maturity").

[b]This column is divided into three subcolumns: "Low" shows the mean percentage of responses that associated the practices in each cluster with level 1 ("No maturity") or 2 ("Somewhat Mature"); "Medium" shows the mean percentage of responses that associated the practices in each cluster with level 3 ("Mature"); and "High", the mean percentage that associated the practices in each cluster with level 4 ("Very Mature") or 5 ("Very High Maturity").

In summary, the analysis of these clusters and practices allowed us to realize that highest maturity in agile software is mainly associated with collaboration, emergent requirements and managing code and tests. The practices based on process areas of CMMI-DEV have still been classified as mature by our respondents, but less mature than others such as collaboration and emerging requirements.

This analysis was based on a closed list of practices. Next section shows how our respondents have freely defined maturity in agile software development: the qualitative results.

5.1.2 Qualitative data analysis

In the content analysis, we identified fifty-two key concepts relative to agile software development maturity, as pointed out by our respondents. These concepts are shown in Table 7. This table also shows the categories of concepts that emerged from the analysis: development practices, process, team, stakeholders, management and outcomes.

Table 7 - Concepts that emerged in the content analysis (FONTANA et al., 2014b)

Category	Subcategory	Concept
Development Practices	-	Configuration Management
		Continuous Delivery of Working Software
		Development Standards
		Sufficient Software Documentation
		Pair Programming
		Refactoring
		Software Testing
		Test-driven Development
Process	-	Application of Agile Practices
		Continuous Improvement
		Definition of Tools and Methods
		Metrics-Based Improvement
		Process Institutionalization
		Standardization of Agile Practices
		Use of Tools and Methods
Team	Knowledge	Keep Lessons Learned
		Knowledge of the Customer's Business
		Knowledge of the Project
		Knowledge of the Technology
		Trained Team
	Behavior	Collaboration
		Commitment
		Making an Effort to Keep Practices in Use
		Self-organization
		Understand Customers
	Communication	Communication within the Team
		Communication with Customers
	Experience	Expertise in Agile Practices
		Time spent working with Agile
Stakeholders	-	Agile Process Acceptance
		Definition of Business Priorities
		Stakeholders Information
Management	-	Definition of Goals
		Process Management
		Process Metrics
		Project Planning

Category	Subcategory	Concept
		Project Tracking
Outcomes	For Management	Efficiency
		Fewer Defects
		Less Effort
		Less Rework
		Less Waste
		Precise Estimates
		Predictability
		Productivity
		Repetition of Results
	For Customer	Delivery on Time
		Effectiveness
		Flexibility
		Generate Value for the Customer
		Product Quality
		Short Delivery Time

The *Development Practices* category shows that practitioners see maturity in practices as related to the way coding is performed, such as configuration management, continuous delivery of working software, development standard, pair programming, testing, among others. The *Process* category shows the concepts that consider maturity in having continuous improvement, definition of tools and methods, process standardization and metrics, all of which are based on agile practices.

The *Team* category groups concepts relative to maturity on the team's *Knowledge*, *Behavior*, *Communication* and *Experience*. The practices listed in this category in Table 7 show that practitioners see maturity on a team that knows the context where it works, has a behavior of collaboration, has commitment and autonomy, communicates efficiently and has experience in agility.

The concepts under the *Stakeholders* category show the importance of the project stakeholders engaging in the agile dynamics. The *Management* category represents the importance of having practices related to planning, tracking and using metrics.

Lastly, the *Outcomes* category represents the concepts that refer to generating results either for the management, or for the customer. Respondents have pointed out that maturity in agile software development may be perceived through the outcomes generated by the team, such as: efficiency, productivity, generating value to customer and product quality, among others.

By analyzing the frequency of occurrence of the concepts and categories, we could identify that the most cited concepts in agile maturity definition are "Product Quality" and "Use of Methods and Tools". The categories that appeared most often in the respondents' definitions were "Outcomes" and "Process", as shown in Figure 14 and Figure 15.

Figure 14 - The concepts by frequency of occurrence (FONTANA et al., 2014b)

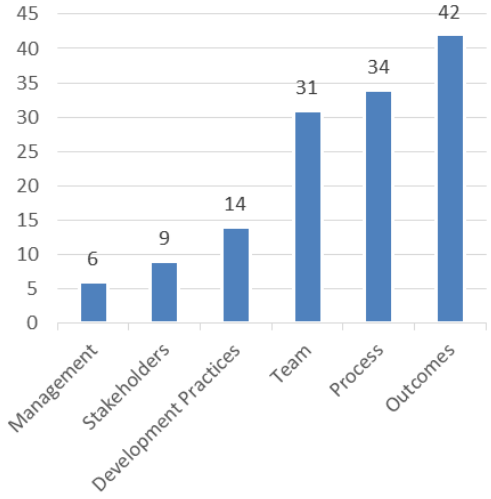


Figure 15 - The categories by frequency of occurrence (FONTANA et al, 2014b)

While codifying and categorizing the responses of the survey, a number of relationships among codes emerged, and we consolidated them on a diagram shown in Figure 16. The team appears as a central category, which applies the processes, is aligned with stakeholders, is directed by the management and applies the development practices. It is the team that generates the outcomes – the concepts more closely related to maturity – for customer and for management.

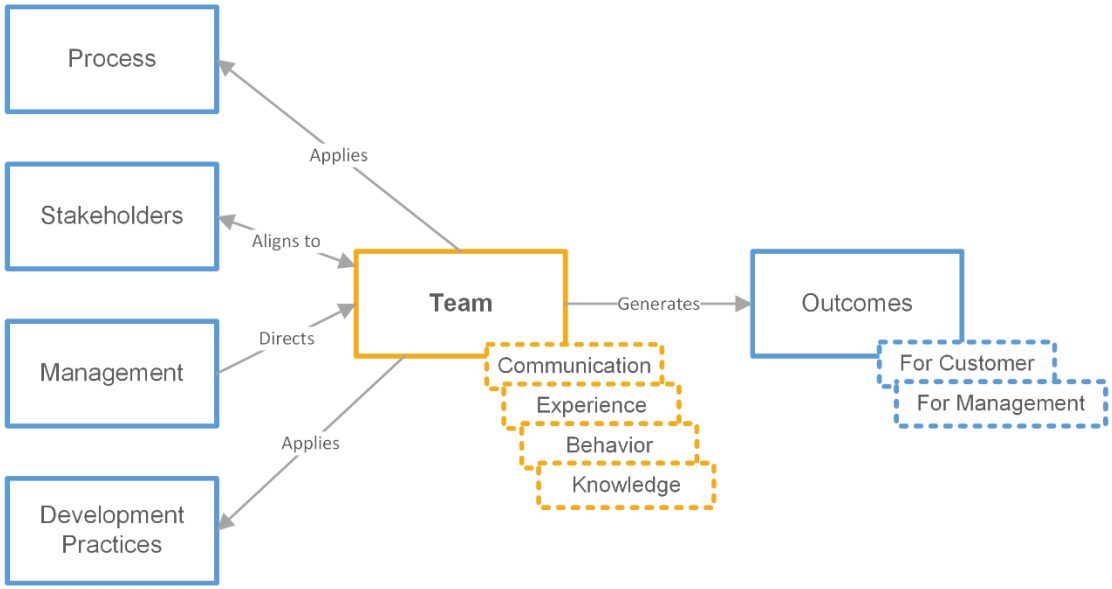


Figure 16 - The relationships among concepts that emerged in content analysis (FONTANA et al., 2014b)

By combining the results we obtained in the analysis of the highest maturity clusters, with the concepts practitioners gave to maturity in agile software development, we could propose the definition presented in the beginning of this section. Next section shows our findings from the investigation on the mechanisms that teams apply to mature in agile.

5.2 How do agile teams get mature?

To have a preliminary answer to this research question, as shown in Section 4.2.1, we surveyed agile practitioners about their perception on the agile maturing process. We received eighty-seven responses from ten different cities in Brazil. The average experience in software engineering of respondents was 10 years, and in agile software development, the average experience was 3.6 years. From all questionnaires, seventy had their practices numbered and the remaining had the practices numbered equally, left blank, or with other comments.

Based mainly on the opinion of experienced practitioners, we could identify that in the agile maturing process, a group of practices should be implemented as a basis: agile values, involved customer, agile planning and agile requirements. Next, other practices could be implemented at an intermediate stage: agile testing and agile coding. Some other practices appeared as relevant to the maturing process, but they could be implemented at any time, such as software architecture, agile physical environment, agile quality assurance and agile project monitoring (FONTANA; REINEHR; MALUCELLI, 2014). As these results are based on practitioners' perceptions, they were used as propositions to the evaluation of evolvement of real agile teams' practices.

To provide an answer to our second research question, we investigated, thus, nine agile teams (as explained in Sections 4.2.2 and 4.2.4). This study comprised interviews, which collected qualitative data about evolvement of agile practices; and questionnaires, which collected quantitative data about ambidexterity and perception of project success. Results of the analysis of the first round of case studies (four first teams) were published in Fontana et al. (2015).

By analyzing the cases, we identified that agile teams mature by pursuing progressive outcomes, in a non-linear and dynamic manner. These outcomes may be accomplished by using a variety of practices and processes, specific to each business context. We represented these outcomes in a framework that we called the

Progressive Outcomes framework, shown in Figure 17. Each line represents a category of outcomes, which emerged from our data analysis: practices learning, team conduct, deliveries pace, features disclosure, software product, customer relationship and organizational support.

Figure 17 - The Progressive Outcomes framework for agile software development maturity

The *practices learning* category comprises the outcomes the teams pursue when they decide to change the way they work. Some of the cases we analyzed, started with an agile trial, in which the team tries adopting agile practices, usually learning “on the fly”, but may not succeed. It evolves to initiatives for agile learning. The teams implement the agile method “by the book” and, with the appropriate

knowledge acquired in practice, it evolves to sensemaking of the work processes. This process includes taking the method learned and tailoring it to particular needs. It is a process based on action, i.e. experimenting, rather than on planning (WEICK; SUTCLIFFE; OBSTFELD, 2005). Later, the teams may start investing in practices to understand what is happening – we named it “comprehension of situation”. It includes, for example, using tools to track the process, having the team to report work status, understanding stories sizes and using simple metrics. This dynamics is similar to the three levels of practice known as the Shu-Ha-Ri distinction (COCKBURN, 2007), which describes that learning processes undergo three phases: repeating the technique “by the book”, then getting autonomy and, later, tailoring the technique based on experience.

The *team conduct* category describes how the team evolves in behavior with the use of agile methods. They start with a responsive behavior, with practices that demand a leadership position of command and control. This team may evolve to a confident team, on which team members start expressing their opinions about the decisions and, later, the assertive team is the one on which the members are active voices in the project and in the process improvement initiatives. This team evolves to a sparkling team: a team that is still assertive, but also characterized by technical excellence, high performance, and motivation to continuous learning.

The *deliveries pace* category describes how the pursued outcomes for deliveries evolve. Teams start investing in iterations to control the coding process: they look forward to having a date to finish the code of a specific requirement. This code is not delivered, it is kept for further testing and integration. The evolvement of this outcome is to implement processes that make this code ready for delivery. On one of the teams included in this study, for example, this initiative was related to implementing a functional test phase that would assure that the software is ready for delivery, but would not be delivered yet. This process of producing ready deliverables evolves, then, to actual deliveries at the end of the iterations – usually late deliveries. The team then, in the next outcome, starts working on practices to have a defined delivery, that is, a delivery that is performed on time.

Features disclosure category describes outcomes teams pursue when defining the features the software will comprise. The requirements gathering outcome represents a process of eliciting requirements similar to those of the traditional software process, with most of requirements being defined at the beginning of the project. It

evolves to practices that allow for requirements discovery, that is, the team starts to iterate the elicitation of requirements, to use stories and to allow requirements to change. Another outcome they pursue is to improve the quality of the requirements to make sure they meet customer expectations. Quite different practices appeared to accomplish this outcome, such as using videos to record customer requirements, using systems analysis diagrams, or involving developers in requirements definition.

The outcomes for the *software product* category describe what the team pursues when it implements the practices to improve the software itself. It starts with a focus on a high-level source code: pair programming and refactoring are examples of practices they perform to have a good resulting source code. We named the next outcome “awareness of failures” because it is when the team realizes that deliveries may have bugs that have to be fixed, and that the processes to accomplish that need to be adjusted. Then, the next step is to focus on high-level delivered software, i.e., ensuring a good delivery. They invest in having a functional tester, or a test team, for example. The last outcome is efficient coding, when practices such as integration, testing and deploy automation are implemented to increase efficiency in the coding process, and sustain the quality that was already achieved with the outcomes that had been previously accomplished.

The *customer relationship* category comprises the outcomes the team pursues when implements practices to improve its relationship with the customer. The first outcome represents the team gaining awareness of the customer, and understanding the customer’s processes and needs. Then, when the relationship evolves, the customer becomes aware of the team, and gets acquainted with the team’s capabilities and the agile work processes. It evolves to a confident customer that knows when the deliveries are going to happen and what is going to be delivered. The customer becomes, then, a partner of the team as there is so much confidence that the team helps define requirements and solutions for the customer’s business problems.

Lastly, the *organizational support* category describes the outcomes related to the position of the organization when providing support for agile transformation. We identified that organizations may start with agile motion when isolated, and small bottom-up initiatives start for agile adoption. It evolves to agile commitment, when top management gains awareness of agile methods and starts supporting the adoption. Next, we identified agile priority as an outcome characterized by the organization that changes its structures, roles and processes to enable the agile transformation. The

last outcome is agile business. The company is recognized for being agile, and agile principles go beyond software development processes.

These results emerged from the analysis performed with the data from the teams we studied, described in the next subsection.

5.2.1 Within-case analysis

The sample of teams in this research includes teams within the four classifications of software services in the Brazilian industry (SOFTEX, 2012): 1) development of software on demand; 2) development of customizable software; 3) development of non-customizable software; and 4) web portals and internet information services. Team sizes range from five to twenty people, company sizes range from seventeen to fifteen thousand employees, and agile adoption time from 0.4 year to six years. The majority of the companies are mid-sized, whose teams have less than ten people, and with agile adoption time ranging from three to six years. Table 8 shows the detailed profile of the teams, and Figure 18 summarizes the variety of the characteristics of the teams in the sample.

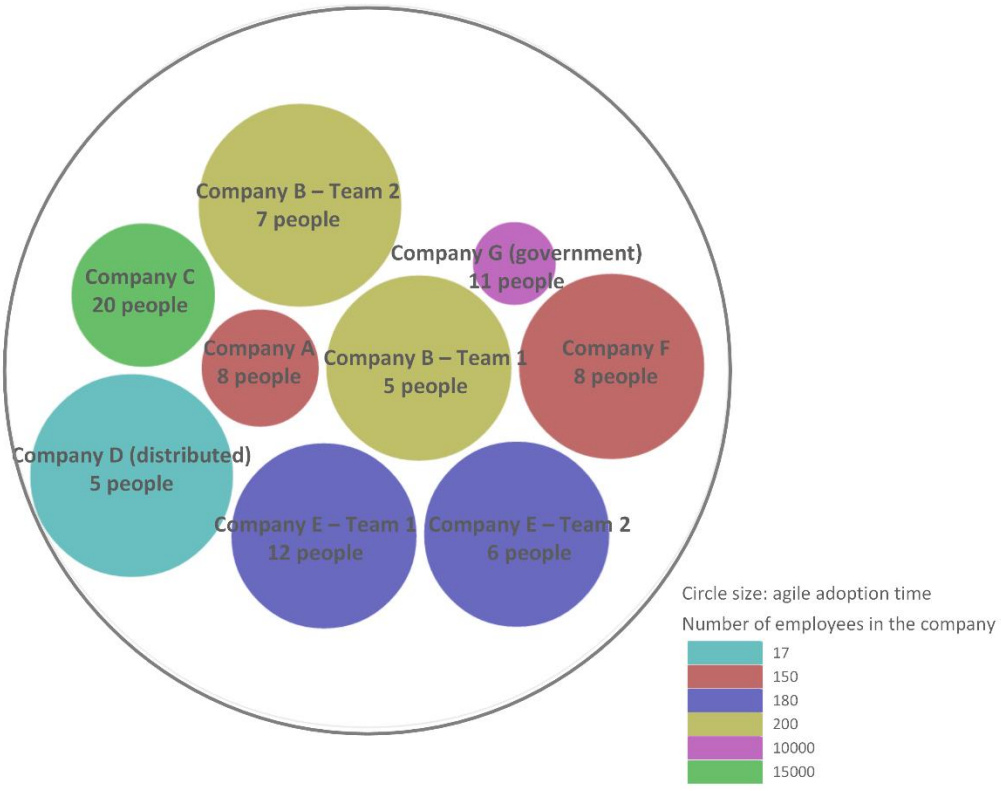


Figure 18 - Variety of the characteristics of the teams in the sample

This section is organized, thus, to show the results of individual teams and, later, the cross-cases analysis, in which the four propositions were evaluated (see Figure 11 on page 54). While performing the studies, we could identify the outcomes each team accomplished according to our framework. They are presented individually in APPENDIX H.

Table 8 - Profile of teams in case studies

Team Alias	Profile	
Company A	Team size	8 people
	Company size	100 people
	Main activity	Documents Management
	Agile adoption time	1 year and 3 months
	Project duration	1 year
	Customers	Inside and outside the company
	Develops software	For its own use and software packages for external customers
Company B – Team 1	Team size	5 people
	Company size	200 people
	Main activity	Educational Technology
	Agile adoption time	5 years
	Project duration	6 months
	Customers	Inside and outside the company (Brazil and South America customers)
	Develops software	On demand
Company B – Team 2	Team size	7
	Company size	200 people
	Main activity	Educational Technology
	Agile adoption time	6 years
	Project duration	1.5 years
	Customers	Inside and outside the company (Brazil, South America, Europe and Asia customers)
	Develops software	Software packages and embedded systems
Company C	Team size	20 people
	Company size	15.000 people
	Main activity	Telecommunications
	Agile adoption time	3 years
	Project duration	6 months
	Customers	Inside the company
	Develops software	Customizes or adapts existing software
Company D	Team size	5 people (distributed)
	Company size	17 people
	Main activity	Web software development
	Agile adoption time	6 years
	Project duration	6 months to 1 year
	Customers	Inside and outside the company (national)
	Develops software	For its own use and software on demand
Company E – Team 1	Team size	12 people
	Company size	180 people
	Main activity	CRM and billing software development
	Agile adoption time	5 years
	Project duration	3 months
	Customers	Outside the company (national)
	Develops software	Customizes or adapts existing software
Company E – Team 2	Team size	6 people
	Company size	180 people
	Main activity	Architecture support for CRM and billing software development
	Agile adoption time	6 years
	Project duration	1 year
	Customers	Inside the company and outside the company (national)
	Develops software	Software for its own use, software on demand
Company F	Team size	8 people
	Company size	150 people
	Main activity	Consultancy and software development
	Agile adoption time	5 years
	Project duration	1 to 2 months
	Customers	Outside the company (North America)

	Develops software	On demand and for its own use
Company G	Team size	11 people
	Company size	10.000 people
	Main activity	Information and communication technology services for public sector
	Agile adoption time	4 months
	Project duration	4 months
	Customers	Outside the company (National)
	Develops software	Customizes or adapts existing software, develops on demand

5.2.1.1 Company A

The team in Company A is responsible for developing and maintaining a single software package that is sold to customers with little customization. They adopted agile methods to help organize their processes, as they used to have an *ad hoc* software development process. Their context is characterized by frequent unplanned requirements. The work processes are currently based on the use of simple tools and, on a daily basis, if they feel the need, they change these processes and experiment new ways of working.

The analysis of the evolvement of agile practices on the team in Company A is summarized in Figure 19. The left-hand box represents the outcomes the team pursued in the past, the box in the middle represents the outcomes the team pursues in the present, and the right-hand box represents the outcomes the team is planning to pursue in the future. These outcomes were inferred, based on characteristics, facts and initiatives we identified in the interviews, which we named *evidence*. The evidence that led us to infer each of the outcomes is described in Table 9.

They started adopting practices to make sense of their work processes. We called it sensemaking because it was an effort to change the current situation and the action was the focus, not the choice (WEICK; SUTCLIFFE; OBSTFELD, 2005). Sensemaking is a process of organizing, but people “make plausible sense retrospectively, while enacting more or less order into […] ongoing circumstances” (*Ibid*., p. 409). “Direct experience is also connected to processes of sensemaking: the combination of a past moment, a connection, and a present moment of experience is what creates a meaningful definition of the present situation.”(DYBÅ; MAIDEN; GLASS, 2014, p. 33)

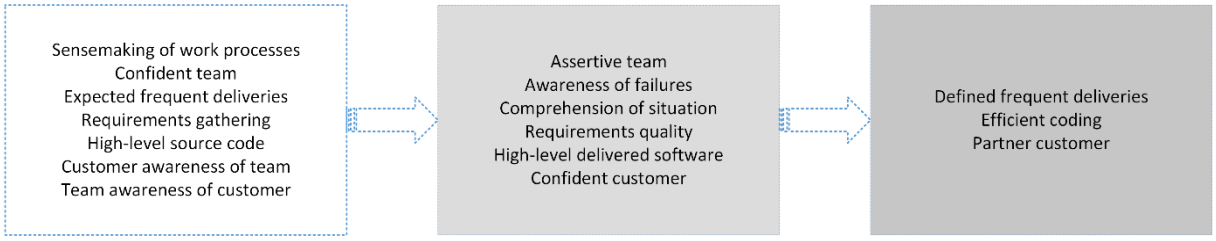


Figure 19 - Analysis of agile practices evolvement in Company A

They also started with a confident team, as they created the team exclusively with experienced people. In the past, they implemented practices to have expected frequent deliveries. They planned for that, but usually delayed the delivery. With respect to requirements, they had a process of requirements gathering, without the concern to be certain of customer needs. Refactoring is a practice they have implemented since the beginning of agile adoption, so we inferred that they pursued a high-level source code. They also invested in increasing the team awareness of customers' needs and business, and vice-versa: the customer gets to know what the team does and how the team works.

In the present, the confident team evolved to an assertive team that influences decisions in the projects and in the processes. They started to perform activities that showed they were aware of their failures, such as performing functional tests; and activities, such as reporting work, so as to enable them to comprehend their situation. They started to focus on requirements quality, implemented practices to have more than a high-level source code, but high-level delivered software instead. The customer, who knew little about team's work (was only aware of the team) in the past, started to become confident about team's deliveries and decisions.

For the future, the team plans to invest in having less delay on deliveries, to make coding more efficient with test automation and, also, to start assisting the customer in defining requirements, creating a partnership with the customer. All the evidence found for each of these outcomes is listed in Table 9.

Table 9 - Evidence for the outcomes identified for the team in Company A

Moment	Outcome	Evidence
Past	Sensemaking of work processes	Implementing the agile method to stop *ad hoc* development; organizing development processes, and adopting tools to support the process dynamics.
	Confident team	Creating an experienced team; knowing closely each person on the team. Assigning the tasks to the team, but having the members define task priorities and estimates.
	Expected frequent deliveries	Planning with sprints to control the coding cycle; estimating correctly.

	High-level source code	Performing pair programming; refactoring; caring about the code.
	Customer awareness of team	Getting the customer to know what is delivered.
	Team awareness of customer	Understanding customers' needs and demands.
Present	Assertive team	Defining politics for the acceptance of unplanned requirements; maintaining a sustainable work pace; feeling confident to deliver the software; enabling the team to change task assignments.
	Failures awareness	Considering time in the sprint for debugging, performing functional and unit tests; looking for improvement in software quality.
	Comprehension of situation	Drawing up thorough plans, reporting work status, planning longer sprints, adopting simple metrics.
	Requirements quality	Improving requirements definition (using recorded videos with the requirements specification).
	High-level delivered software	Including a testing phase in the sprint; assuring that maintenance does not create new bugs; Improving code version control.
	Confident customer	Formalizing new requirements orders; feeling customer trust; controlling requirements cycle; reducing delivered bugs.
Future	Defined frequent deliveries	Not delaying deliveries.
	Efficient coding	Automating unit tests.
	Partner customer	Hypothesizing customer needs; defining the roles and rights of the information technology department.

We also analyzed the perception that the team had on their ambidexterity. Table 10 shows the percentage of the team that disagreed, that were neutral, or that agreed on the perception of performance, good alignment, bad alignment, adaptability, bridging ties and strong ties. It also shows the p-values for the Chi-Square tests and the rejection or not of the null hypothesis. In Company A, the team has a consolidated view of their ambidexterity abilities, as all of the hypothesis were rejected, and the percentages of agreement and disagreement indicate positive evaluations on management alignment and adaptability.

Project success perception in Company A is shown in Table 11. Each of the factors was evaluated by the team leader as (1) meaning no agreement to have reached the benefit through (5) meaning full agreement to have reached the benefit. The data showed that the only factor they have not realized with adoption of agile methods was "Improved Business Processes". Table 11 shows the results in Company A compared with the means from the other teams and standard deviations. The means for the other teams were calculated based on the responses given to each factor by the other teams, without considering Company A's responses.

Table 10 - Ambidexterity data for Company A

	Performance	Good Alignment	Bad Alignment	Adaptability	Bridging Ties	Strong Ties
Disagree	0.0%	0.0%	87.5%	0.0%	0.0%	0.0%
Neutral	12.5%	0.0%	6.3%	8.3%	0.0%	5.0%
Agree	87.5%	100.0%	6.3%	91.7%	100.0%	95.0%
p-value	0.000	0.043	0.001	0.000	0.000	0.000
Rejects H0	Yes	Yes	Yes	Yes	Yes	Yes

Table 11 - Projects success perception in Company A

	Company A	Means for Other Teams	Std Dev
Reduced delivery schedules	4	5.0	0.0
Increased return on investment (ROI)	5	3.3	0.6
Increased ability to meet current customer requirements	4	3.7	0.6
Increased flexibility to meet changing customer requirements	5	3.7	1.5
Improved business processes	3	4.0	1.0

As a summary, agile evolvement in Company A was not associated with increasing agility. In this context, Team A evolved practices for customer relationship and the quality of the requirements, the code and the software. By associating the ambidexterity results with project success, we see a team with ambidextrous abilities that is satisfied with the results they achieve with their agile projects.

5.2.1.2 Company B – Team 1

This is the first team from Company B, where we performed two case studies. This team develops software to respond to government requirements in educational technology. Thus, each project develops a completely new product, with new technologies, to unknown customers and unknown users. They have an interdisciplinary team and the support to users is performed by other people. They adopted the agile method based on a top-management decision and, before that they had an ad hoc software development process. They use simple tools to support the work processes. If the project demands, they change these work processes or even abandon them in order to be faster.

In the analysis of the agile practices evolvement, shown in Figure 20, we could realize that they started with a confident team that defined their own priorities. This team evolved to an assertive team, which self-organized and changed processes as needed. As they were told to use agile methods, they went through a process of agile learning, which evolved to practices that aimed at sensemaking the work processes. Since the beginning of agile adoption, they have been using iterations to finish the code. We called it “Expected frequent finished coding”, as they do not have to deliver. They just use the sprints dynamics to control the targets for finishing the source code. In the past, they already had practices to allow for requirements discovery, that is, they discovered the requirements iteratively. Since the beginning, they also had to implement practices so that a third party can support the software in production.

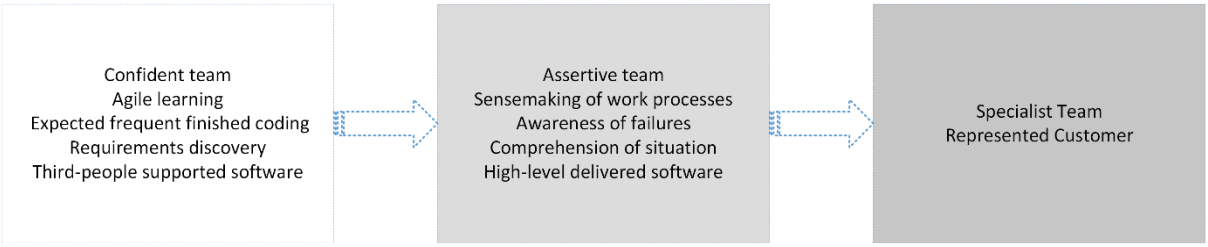


Figure 20 - Analysis of agile practices evolvement in Company B – Team 1

In the present, they also implement practices that show their awareness of failures and their need to comprehend their situation, as well as have high-level delivered software. In the future, this team will abandon agile methods, due to an organizational re-engineering. Thus, they will have a Specialist Team, on which roles will be separated and the customer will be represented by a specific department in the company. Table 12 shows the evidence we found for these outcomes.

Table 12 - Evidence for the outcomes identified for Team 1 in Company B

Moment	Outcome	Evidence
Past	Confident team	Creating an experienced team; knowing each person on the team closely; having the team define tasks and priorities.
	Agile learning	Following an agile method “by the book”.
	Expected frequent coding finished	Not delivering at the end of the sprint; testing the software after the sprint has finished.
	Third-people supportable software	Creating documentation at the end of the process; using text documents to define requirements.
Present	Assertive team	Allowing the team to self-organize; having the team define tasks and priorities; playing to win.
	Sensemaking of work processes	Tailoring the agile method, for example, increasing sprint size.
	Awareness of failures	Considering time in the sprint for debugging.
	Comprehension of situation	Drawing up thorough plans; defining requirements iteratively.
	High-level delivered software	Performing unit and functional tests; applying tools to support development.
Future	Specialist team	Assigning a role to each team; formalizing software architecture definition; having a team perform functional tests formally.
	Represented customer	Having a formal structure to represent customer.

In the ambidexterity data analysis of Team 1 in Company B (Table 13), we could realize that the team does not have a consolidated perception of ambidextrous abilities. Although the hypothesis for performance, bad alignment, bridging ties and strong ties was rejected, the good alignment and adaptability perception on the team is not a consensus. It means that the team feels that it receives conflicting objectives, wasting resources on unproductive activities and that the work processes are not flexible enough to engage the team into innovative solutions.

Table 13 - Ambidexterity data for Team 1 in Company B

	Performance	Good Alignment	Bad Alignment	Adaptability	Bridging Ties	Strong Ties
Disagree	15.0%	40.0%	20.0%	60.0%	0.0%	0.0%
Neutral	5.0%	20.0%	0.0%	20.0%	0.0%	0.0%
Agree	80.0%	40.0%	80.0%	20.0%	100.0%	100.0%
p-value	0.001	1.000	0.030	0.223	0.000	0.000
Rejects H0	Yes	No	Yes	No	Yes	Yes

Although Team 1 in Company B has presented negative results in the ambidexterity perception, project success perception was similar to the highest value in all the factors. Table 14 presents the values for this team, the means of other teams and the standard deviation.

Table 14 - Projects success perception on Team 1 - Company B

	Company B – Team 1	Means for Other Teams	Std Dev
Reduced delivery schedules	5	4.7	0.6
Increased return on investment (ROI)	4	3.7	1.2
Increased ability to meet current customer requirements	4	3.7	0.6
Increased flexibility to meet changing customer requirements	5	3.7	1.5
Improved business processes	5	3.3	0.6

We conclude that Company B – Team 1 does not need to deliver continuously, but they still used and evolved agile methods to control coding and requirements. The agile evolvement was mainly on learning and, later, sensemaking the process, as well as evolving the team. There is a contradiction when we see that, although project success perception is good, the team does not agree on ambidextrous abilities. In the validation presentation, the team leader explained that this contradiction is due to the fact that the members felt confident inside the team and about their individual results, but lacked the support of the organization as whole.

5.2.1.3 Company B – Team 2

Team 2 in Company B has a different context from that of Team 1. They are a team for new products development. They define their own requirements for the software products they build, which are validated by a team of managers that represent the customer. This is the reason why they have an interdisciplinary team. The work processes are supported by simple tools and they have a defined process (mostly based on Scrum) that changes eventually if needed.

Figure 21 shows the evolvement of agile practices in Team 2, Company B. This team started with practices focusing on Agile Learning, as they adopted the agile method based on a top-management decision. As agile practices were learned, they

could implement practices aiming at the sensemaking of work processes, to tailor the agile method and change the work processes to fit their needs. In the beginning of agile adoption, they had a team separated by roles, which we called Specialist Team. It soon evolved to an interdisciplinary team with responsive characteristics, that is, which did what they were told to do. The efforts on the development of the team (including firing people that did not have the agility skill) led the team to become a confident team. Now, they are considered to have an Assertive Team.

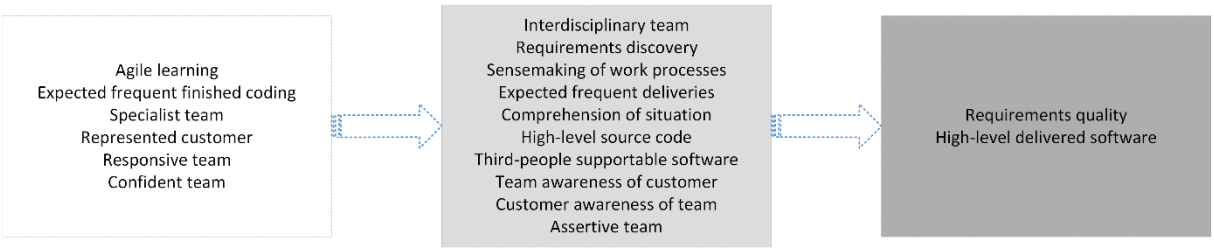


Figure 21 - Analysis of agile practices evolvement in Company B – Team 2

In the present, this team also focuses on having requirements discovery, comprehension of situation and high-level source code. Iterations, which in the past controlled the coding process, now focus on delivering working software. They are now investing in creating means for the team to be aware of the types of demands of the managers who represent customer, and for these managers to be aware of the team's work processes. In the future, this team plans to work on requirements quality, by using systems analysis diagrams to define the software; and on high-level delivered software, as they plan to invest in software testing. Table 15 shows the evidence we found in the case for each of the outcomes.

Table 15 - Evidence for the outcomes identified for Team 2 in Company B

Moment	Outcome	Evidence
Past	Agile learning	Following an agile method "by the book".
	Expected frequent coding finished	Not delivering at the end of the sprint.
	Specialist team	Having the functional tests performed on a separate team; Having the roles separated by different teams.
	Represented customer	Having the top management define requirements priorities.
	Responsive Team	Having responsive people who later left the team.
	Confident Team	Hiring confident people.
Present	Interdisciplinary team	Having a team with multiple profiles; sharing knowledge
	Requirements discovery	Validating requirements with customer; having the developers become acquainted with the requirements before starting work; formalizing architecture definition, having sprints to deliver documentation.
	Sensemaking of work processes	Organizing work processes; tailoring the agile method.
	Expected frequent deliveries	Delivering (late) at the end of the sprint.
	Comprehension of situation	Monitoring the project formally, with the help of a simple tool; having a better cost estimation.
	High-level source code	Integrating code daily.
	Third-people supportable software	Creating and updating documentation.

	Assertive team	Having the team to help define requirements.
	Team awareness of customer	Designing and publishing a work process.
	Customer awareness of team	Designing and publishing a work process.
Future	Requirements quality	Improving requirements definition (using UML), performing documentation review, prototyping before development.
	High-level delivered software	Automating functional tests, implementing test driven-development, having a team to test.

The ambidexterity data for Team 2 in Company B presents a consolidated perception on the team about performance, bridging ties and strong ties, but problems with alignment and adaptability perception. The null hypothesis was not rejected for perception of performance, but almost a third of the team reported that they did not have a formed opinion. Table 16 shows that the null hypothesis could not be rejected for good alignment, bad alignment and adaptability.

Table 16 - Ambidexterity data for Team 2 in Company B

	Performance	Good Alignment	Bad Alignment	Adaptability	Bridging Ties	Strong Ties
Disagree	3.6%	0.0%	35.7%	28.6%	0.0%	0.0%
Neutral	28.6%	42.9%	7.1%	9.5%	0.0%	5.7%
Agree	67.9%	57.1%	57.1%	61.9%	100.0%	94.3%
p-value	0.000	0.076	0.324	0.113	0.000	0.000
Rejects H0	Yes	No	No	No	Yes	Yes

Project success perception presents a team that perceived less return on investment and less ability to meet customer requirements with agile adoption, in comparison to the others. On the other hand, reduced delivery schedules had a good rating, as well as flexibility to meet changing customer needs and improved business processes. Table 17 shows the data for this team, as well as the means of other teams and standard deviations.

Table 17 - Project success perception in Team 2 - Company B

	Company B – Team 2	Means of Other Teams	Std Dev
Reduced delivery schedules	5	4.7	0.6
Increased return on investment (ROI)	3	4.0	1.0
Increased ability to meet current customer requirements	3	4.0	0.0
Increased flexibility to meet changing customer requirements	4	4.0	1.7
Improved business processes	4	3.7	1.2

As a summary, Team 2 in Company B started agile adoption from a top-management initiative. They initially focused on implementing the agile method “by the book” and on controlling their coding cycles with iterations. They evolved to have actual deliveries at the end of the iterations and to focus on a high-level source code. The team clearly started as a responsive one, and became confident. Currently, they are active actors in the projects. The initiatives to have the customer (i.e. the managers

who represent the customer) to understand the team and vice-versa were not implemented at the beginning of agile adoption, but at a later stage. Ambidexterity analysis did not perform well, but project success has been identified mainly by reduced delivery schedules.

5.2.1.4 Company C

Company C is a big telecommunications company, with a rigid structure. The team we investigated is responsible for developing customizations for the information technology platform of the company. The software packages are customized to meet customer requirements and, thus, the company's strategies. Before agile adoption, the company had a traditional software development process. The context of the team is characterized by very strict delivery dates and the roles are separated into different hierarchical structures. To develop software projects, people are temporarily placed physically together. The work processes are defined and hard to change, so few adaptations are performed when necessary. Both developers and managers use a variety of tools to support their work.

Figure 22 shows the summary of the evolution of the outcomes in Company C. They started adopting agile methods as a sensemaking of work processes; they used the agile "by the book", with adaptations to the rigid structure they have. Iterations, for example, were focused on having finished code, and not on deliveries. As they had a traditional software process, the practices that enabled comprehension of their situation were already performed. The requirements definition process was based on heavyweight documentation and some initiatives focused on having the customer to be aware of the team's processes.

From the organizational point of view, some years ago, the company went through an agile motion, in which some isolated small initiatives happened in some teams. Agile adoption has officially begun with agile commitment, when top management has supported and incentivized agile transformation.

At present, this team is implementing practices that show they are working on the awareness of their failures, such as considering time in the sprint planning to correct bugs from previous sprints. The requirements definition activities now allow for requirements discovery, with user stories. Iterations, which used to focus on having the code finished, now create deliverables. The difference lies in the fact that now the solution is tested and gets ready to go to production, but they do not deliver, given the

difficulty in delivering partial customizations of the software platforms. With respect to the team, the team leader pointed out they are now facing an issue with a responsive team. Development staff does not engage in daily meetings, and does not see the practices as relevant. They are also focusing on high-level delivered software and on initiatives to make the customer confident. From the organizational point of view, evidence shows agile is a priority, with teams' structural changes to ease communication and training sessions throughout the company.

Figure 22 - Analysis of agile practices evolvement in Company C

In the future, they plan to work on requirements quality, on enhancing team confidence and on the standardization of the agile practices across the teams. Table 18 shows the evidence we found for the outcomes pursued by the team in Company C.

Table 18 - Evidence for the outcomes identified for the team in Company C

Moment	Outcome	Evidence
Past	Sensemaking of work processes	Following an agile method "by the book", but having each team use agile its own way.
	Expected frequent coding finished	Drawing up plans with sprints; having variable sprint sizes; integrating code by implemented feature.
	Requirements gathering	Having a heavyweight text documentation of requirements.
	Customer awareness of team	Performing user acceptance test at the end of the project.
	Comprehension of situation	Developing project scope using work breakdown structure diagrams; starting project only with perceived scope maturity; reporting work status; having people to help on administrative tasks.
	Agile motion	Isolated small initiatives
	Agile commitment	Top management involvement; training with management staff; pilot projects; resistance to agile adoption
Present	Awareness of failures	Considering time in the sprint for debugging and production support.
	Requirements discovery	Defining requirements based on a vague product description; including an analysis phase in the project; having the requirements defined as stories and versioning stories.
	Expected frequent deliverables	Including a testing phase in the sprint; identifying a minimum releasable product, but not delivering at the end of the sprint.
	Responsive team	Having the team define personal tasks, but macro plan is defined by the software architects; top management listens to suggestions for improvement, but not necessarily considers them.
	High-level delivered software	Including a testing phase in the sprint, implementing tools to automate integration of code in different environments.

	Confident customer	Helping customer to define business value.
	Agile priority	IT structure has changed to ease communication; there is a roadmap but not all teams work the same way; everybody was trained (overview or specific courses); awareness that agile methods need adaptation for big companies; dealing with the resistance of some people
Future	Requirements quality	Improving product description.
	Standardization of agile practices	Making agile practices similar across teams.
	Confident team	Development team does not take responsibility for the project.

Ambidexterity data for the team in Company C showed agreement on performance perception, although almost a third of the team has a neutral opinion. For the good alignment, bad alignment and adaptability evaluations, we could not reject the null hypothesis, which reflects disagreement among team members. At the validation presentation, the team leader was concerned about the 83.3% rate of neutral opinions about good alignment and the issue was taken to the top management. They justified it as a problem of cultural change and resistance to agile methods, different forms of application of agile methods around the company and too many processes that are still hard to change because of established applications and procedures. The Chi-Square test for bridging ties and strong ties rejected the null hypothesis, showing agreement about ties on the team. Table 19 shows the details for these data.

Table 19 - Ambidexterity data for the team in Company C

	Performance	Good Alignment	Bad Alignment	Adaptability	Bridging Ties	Strong Ties
Disagree	8.3%	0.0%	41.7%	38.9%	0.0%	3.3%
Neutral	33.3%	83.3%	16.7%	33.3%	11.1%	6.7%
Agree	58.3%	16.7%	41.7%	27.8%	88.9%	90.0%
p-value	0.006	0.131	0.959	0.320	0.000	0.000
Rejects H0	Yes	No	No	No	Yes	Yes

Projects success perception in Company C, shown in Table 20, presented three factors evaluated as below the means of the other teams: they felt less return on investment, less flexibility to meet changing customer requirements and fewer improved business processes. The greatest benefit they realized with the agile adoption was reduced delivery schedules.

Table 20 - Projects success perception in the team in Company C

	Company C	Means for Other Teams	Std Dev
Reduced delivery schedules	5	4.7	0.6
Increased return on investment (ROI)	3	4.0	1.0
Increased ability to meet current customer requirements	4	3.7	0.6
Increased flexibility to meet changing customer requirements	2	4.7	0.6
Improved business processes	3	4.0	1.0

The investigation of the team in Company C showed a scenario where a change in culture was necessary for agile adoption. The team started pursuing the sensemaking of work processes, with the iterations to control the coding process. The initiatives evolved to pursue outcomes on requirements, customer confidence and high-level deliverable software. The issue with this team was the members' responsive attitude, which reflected the bad results of the ambidexterity analysis. From all the teams we analyzed, this was the one with lowest perception of project success.

5.2.1.5 Company D

This company was created with the purpose of using agile methods. They develop software exclusively for the web and the team we analyzed customized a software they had developed for a specific customer. The team is responsible for developing and providing support to the software in operation. They have a distributed team, in two different cities in Brazil. They started using agile methods trying to learn "on the fly", without training, but did not succeed. From this moment on, they always endeavor high-quality training when needed and focus on learning from practice. This team relies on tools to support communication and project monitoring.

Figure 23 shows how the outcomes evolved on this team, and Table 21 shows evidence for each outcome. Most of them were placed in the past because, since the company was created, the team has already improved its agile practices. As we mentioned, the team started with an agile trial. At that time, they did not even know how to deliver value to the customer. This outcome evolved to agile learning, when the team received a number of high-quality training sessions in agile requirements and agile engineering. Now, this outcome evolved to sensemaking of work processes. It is characterized by a moment when the team knows the value of each practice. They are, therefore, capable of abandoning practices or re-adopting them as needed. They used to have a young, responsive team, which evolved to a confident team who knew the process and helped customer defining requirements. Today they have an assertive team, which stopped using estimates, and has made work processes more flexible.

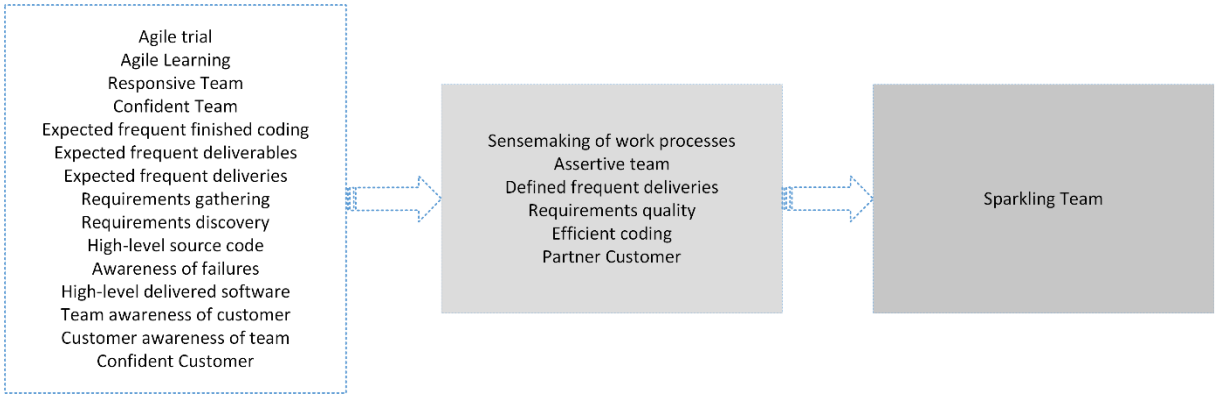


Figure 23 - Analysis of agile practices evolvement in Company D

Today this team performs continuous deliveries to customer, and even delivers before customers expect – an outcome we refer to as defined frequent deliveries. To achieve this outcome, the team has learned from iterations that just finished code (expected frequent finished coding), iterations that created deliverables but did not actually deliver to customer (expected frequent deliverables), and late deliveries (expected frequent deliveries). In the past, they had already focused on high-level source code, by applying pair programming, for example. They faced a moment of awareness of failures, when they realized it was not enough to deliver quality and, then, they started focusing on high-level delivered software (see evidence in Table 21). Now, they have an efficient coding, with a complete automated environment for integration, testing and deploying.

As the team has a single customer, the evolvement of this relationship was clear in the interviews. The customer learned with the team, participating actively in meetings and even receiving training from the Scrum Master. Table 21 shows the evidence we found for the evolution from team awareness of customer, and vice-versa, to a confident customer and, now to a partner customer.

Now they are facing problems of conflicts in the team and less willingness to learn, so we assume that, for future, they will work on practices to create a sparkling team.

Table 21 - Evidence for the outcomes identified for the team in Company D

Moment	Outcome	Evidence
Past	Agile trial	Implementing agile on the fly, Don't know how to deliver value, Badly defined requirements
	Agile Learning	Use agile "by the book", Got the team trained in agile requirements/engineering
	Responsive Team	Scrum Master role well-defined as a motivator and facilitator, No policy for accepting extra demands, Delay because of extra demands

	Confident Team	After training, team helped defining requirements with customer (Customer could not fill a sprint with requirements), Improve work environment for enhanced communication, Team became distributed (Communication effort, Virtual kanban to know what is happening), No rules for branching code, Natural code standard (Collective ownership of the code), Team performs business analysis with customer
	Expected frequent finished coding	Code was ready, but they didn't know how to deliver value
	Expected frequent deliverables	Manual and slow environment configuration and deploy, Estimates with story points
	Expected frequent deliveries	Delays because of extra demands
	Requirements gathering	Badly defined requirements
	Requirements discovery	Requirements based on product backlog (Short stories, loose goals for the sprint, backlog grooming when story is chosen, acceptance criteria to user stories)
	High-level source code	Pair-programming, Cleaning-code, Refactoring during stories build
	Awareness of failures	Bugs from previous sprint to correct, Manual and slow environment configuration and deploy
	High-level delivered software	Got the team trained for engineering techniques (XP),Test-driven development, Simple design, Informal code review, Manual functional test, Few Bugs, eventual environment problems
	Team awareness of customer	Got the team training for agile requirements
	Customer awareness of team	Get the customer trained in agile processes (requirements, prioritization...)
	Confident Customer	Physical task board and virtual Kanban, After training, team helped defining requirements with customer, Defined politics to accept extra demands, Meetings take customer attention, Customer knows what fits in a sprint, Customer is aware of his role in the process
Present	Sensemaking of work processes	Flexible process to define tasks, Team decides when to use physical task board, Few metrics (Number of items in the sprint, Number of extra items in the sprint)
	Assertive team	Informally Control WIP, Defined politics to accept extra demands, Team helps defining requirements, Loose goals for the sprint, Stopped using estimates, Team defines tasks (No more need to break stories into tasks, Flexible process to define tasks), Team decides when to use physical task board
	Defined frequent deliveries	Deliver before sprint finishes, Continuous delivery, Short stories, Loose goals for the sprint, Backlog grooming when story is chosen, Acceptance criteria to user stories, Set stories sizes to standards
	Requirements quality	Requirements based on product backlog (Short stories, loose goals for the sprint, backlog grooming when story is chosen, acceptance criteria to user stories)
	Efficient coding	Devops infrastructure, Automated integration (Ease work environment developing its own tools), Fast bugs correction, Continuous delivery (Sprints are kept for customer awareness)
	Partner customer	Release planning with business analysis (Release planning with business analysis, Stories defined based on business analysis), Support the customers in his/her business (training team to customer development)
Future	Sparkling team	Challenges on recycling knowledge (conflicts with team members opinions, lost team motivation for learning), re-collocate team

The ambidexterity analysis for this team (Table 22) shows – regarding performance, adaptability, bridging ties and strong ties – that the team has the same perception, and the percentages show good results. Regarding alignment, the null hypothesis could not be rejected, which shows the team lacks a consolidated view of management alignment with business goals. In the validation presentation we did with

the team leader, he explained this result clearly shows his perception: the company is experiencing a change in business strategy and the team may indeed feel that there is lack of alignment.

Table 22 - Ambidexterity data for Company D

	Performance	Good Alignment	Bad Alignment	Adaptability	Bridging Ties	Strong Ties
Disagree	0.0%	0.0%	75.0%	0.0%	0.0%	0.0%
Neutral	12.5%	25.0%	12.5%	8.3%	16.7%	0.0%
Agree	87.5%	75.0%	12.5%	91.7%	83.3%	100.0%
p-value	0.000	0.238	0.123	0.001	0.005	0.000
Rejects H0	Yes	No	No	Yes	Yes	Yes

Regarding project success perception, this team shows the lowest result with reduced delivered schedule. In the validation interview, the team leader explained that it is because they have always had reduced delivered schedules, since they have always been agile, so it was not a perceived benefit. Table 23 presents the values for project success perception on this team in comparison with others means and standard deviations.

Table 23 - Projects success perception in Company D

	Company C	Means for Other Teams	Std Dev
Reduced delivery schedules	3	4.4	0.9
Increased return on investment (ROI)	5	4.0	0.8
Increased ability to meet current customer requirements	4	4.0	0.5
Increased flexibility to meet changing customer requirements	5	4.1	1.1
Improved business processes	5	4.0	0.9

In summary, the team in Company D has been using agile methods for six years, i.e. longer than the Brazilian average (MELO et al., 2013). Their processes are already established and relaxed with the experience of the team. They have a number of tools that support a complete automated environment, which is essential to allow continuous delivery. The challenge is now working on the team, to regain motivation to continue improving.

5.2.1.6 Company E – Team 1

This team has been using agile methods since the company started, five years ago. They develop and maintain a single product for a few customers. These customers are big companies and the team we studied dealt with a customer that has a formal process of creating demands. Thus, requirements arrive mostly defined and they usually do not change. New tasks may arise because of requirements refinement. Delivery schedule is fixed, every two months, with update patches every fifteen days.

Over five years, this company has grown and agile practices have been adapted for big teams, as described in our experience report published in Walter et al. (2015). The company has a team to develop tools to support work processes, which we also studied and described in Section 5.2.1.7.

Figure 24 shows the evolvement of the outcomes that we identified this team pursued. As the team began using agile methods some years ago, a number of practices have been already established since then. In the past, they focused on agile learning and, after that, on sensemaking the work processes. Based on this tailoring of agile practices, they later implemented practices to reach comprehension of the situation, all of them described on the evidence in Table 24. The team, as in other cases that we have studied, started as responsive, with inexperienced developers. Over time, they evolved to a confident team and, in the present, we could characterize them as an assertive team.

Figure 24 - Analysis of agile practices evolvement in Company E – Team 1

They started adopting agile planning with sprints, which we called expected frequent deliverables. However, they soon realized that a continuous flow of tasks would be more adequate to their needs (as described in Walter et al., 2015). We understood it was applied to get to expected frequent deliveries and, later, to defined frequent deliveries. Requirements have always been allowed to change, and now practices are being implemented to focus on quality improvement, by involving developers on stories definition. Pair programming and review, among other practices described in Table 24, contributed to a focus on high-level source code. They have experienced awareness of failures when they sensed problems with code integration. The investment on tests automation (they have more than 30.000 automated tests) shows a focus on a high-level delivered software. They have always had a confident customer.

In the future – some practices have already begun – evidence shows this team is investing on creating a motivating environment to foster a sparkling team. They are also planning to invest in test-driven development and other practices to accelerate development to get to an efficient coding. Table 24 shows the evidence we found for each of these outcomes.

Table 24 - Evidence for the outcomes identified for the team in Company E - Team 1

Moment	Outcome	Evidence
Past	Agile learning	Planning with sprints; physical task board and issue tracker (sync problems); use pair programming; user stories; planning poker
	Sensemaking of work processes	Started pair rotation; ineffective planning (sprint planning took too much time, estimates are used for commercial commitment, using velocity to adjust estimates just for customer); continuous flow of tasks
	Comprehension of situation	Measuring team velocity; using T-shirt sizes for estimation
	Responsive team	Developers focused exclusively on coding; team working overtime frequently; requirements defined by the specifier with customer (developers not involved in estimation and requirements); code review performed only by the coach (inexperienced people)
	Confident team	Cross-pair review; stimulation of different pairing combinations; tasks usually assigned by the team leader, but new ones are allowed to emerge; developers define build design with technical leader; no specific format for requirements.
	Expected frequent deliverables	Planning with sprints
	Expected frequent deliveries	Continuous flow of tasks
	Defined frequent deliveries	Delivery schedule defined by customer (2 months/15 days); deliveries before deadline
	Requirements discovery	No specific format for requirements; identification of minimum releasable product; requirements are refined only after customer approval; new tasks are allowed to emerge during development due to requirements refinement
	High-level source code	Code review by the coach; having an architecture support team; pair programming; defining a best-practices checklist; people are given time and are encouraged to study coding techniques
	High-level delivered software	Automated functional tests; having a team to execute manual functional tests; caring for software quality; having a team to control branches and releases; attention to bugs; little recurrence of bugs
	Awareness of failures	Problems with code integration (spend more time integrating than coding); cross-pair review (coach review not enough)
	Confident customer	Project manager and specifier close to customer; hardly ever delays deliveries; working with IT department to deploy solution
Present	Assertive team	Flexible WIP size, which allows for pairing “on demand”; change in teams structure to reduce team size (multifunctional teams, sense of purpose to developers, increase team autonomy); high-caliber team
	Requirements quality	Involving developers in requirements elicitation (get people to know what to do)
Future	Sparkling team	Using gamification to motivate team; more celebration on accomplishments
	Efficient coding	Test-driven development; manual end-to-end tests are executed by test analyst for complex builds (find bugs earlier); reduce the need to give first-level support to production; involving developers in requirements elicitation

Ambidexterity perception for this team seems to be aligned with all the aspects we evaluated, as none of the null hypothesis has been rejected. Besides the agreement of the team on the perception of performance, alignment, adaptability and ties, the percentages are positive, as shown in Table 24. The only aspect with less positive evaluation was adaptability.

Table 25 - Ambidexterity data for Company E - Team 1

	Performance	Good Alignment	Bad Alignment	Adaptability	Bridging Ties	Strong Ties
Disagree	0.0%	0.0%	85.0%	6.7%	3.3%	0.0%
Neutral	7.5%	10.0%	10.0%	43.3%	13.3%	8.0%
Agree	92.5%	90.0%	5.0%	50.0%	83.3%	92.0%
p-value	0.000	0.005	0.000	0.000	0.000	0.000
Rejects H0	Yes	Yes	Yes	Yes	Yes	Yes

With respect to project success perception, this team has shown lower evaluation scores than the average of the other teams for reduced delivery schedules and flexibility to meet with changing customer requirements, as shown in Table 26.

Table 26 - Projects success perception in Company E – Team 1

	Company C	Means for Other Teams	Std Dev
Reduced delivery schedules	3	4.4	0.9
Increased return on investment (ROI)	4	4.1	0.8
Increased ability to meet current customer requirements	4	4.0	0.5
Increased flexibility to meet changing customer requirements	3	4.4	1.1
Improved business processes	4	4.1	1.0

Analysis of this team has shown lower projects success perception, but good results for ambidexterity analysis. This team is characterized by having established agile processes and a high level of tests automation, which assures continuous delivery to a big customer company. Challenges in agile evolvement are focused on creating and maintaining a sparkling team and a more efficient coding activity.

5.2.1.7 Company E – Team 2

This team is responsible for applications servers infrastructure and for defining the architecture of the main software product sold by the company. Their responsibility is also to develop tools to automate the software development work environment (build, integration, testing, packaging and deploying). This team consists of experienced people with a focus on applying up-to-date technology. Their projects involve a lot of experimentation – objectives are defined during the project – and they usually lack deadlines. It means that progress is not measured by amount of reported hours but by results (commits or running automated tests).

Figure 25 shows the evolvement of the outcomes we identified on this team. Agile adoption started six years ago with focus on agile learning while using Scrum “by the book”. They experienced the sensemaking of work processes by tailoring practices, according to the different types of projects they perform. The team evolved from a responsive team (with partial autonomy) to a confident team. They started agile adoption drawing up sprints, with expected frequent deliverables, and soon realized that the continuous flow of tasks would lead them to defined frequent deliveries. They clearly have a process of requirements discovery to allow innovation during the project.

Figure 25 - Analysis of agile practices evolvement in Company E – Team 2

The software products they develop have a high-level source code, with pair programming; but they also have a high-level on delivery, since tests automation has been a regular practice since this team was formed. Currently, they have a culture of always creating tools to improve – and continuously automate – work environment, which lead to efficient coding.

With known practices and automation culture, for the future the focus is on the team: resolving conflicts, improving communication and providing sense of purpose and more autonomy to the team. Table 27 shows the evidence for the outcomes described in Figure 25.

Table 27 - Evidence for the outcomes identified for the team in Company E – Team 2

Moment	Outcome	Evidence
Past	Agile learning	Adopting Scrum by the book; Used to estimate tasks, Manual Kanban, Mandatory pair programming
	Sensemaking of work processes	Virtual kanban (not visible all the time), Demands have customized processes, Work processes are more flexible, less bureaucratic, No estimation, no deadlines, Using spikes when needed
	Responsive team	Team had partial autonomy to discuss design; Mandatory pair programming; Retrospectives are done only when bad things happen
	Expected frequent deliveries	Using sprints for planning
	Defined frequent deliveries	Continuous flow of tasks; Progress is measured with commits, builds put in alpha test
	Requirements discovery	New tasks are allowed to emerge during build; Demands come from several different sources and are prioritized; Objectives are refined during the project (use spykes)

	High-level source code	Deploy was not automated; Refactoring done with regular builds; Pair programming
	High-level delivered software	Functional Tests Automation culture; Tests are always developed during the tasks; Frequent code commits (more than once a day); Continuous automated integration
Present	Confident team	Free to identify opportunities for automation; Team is not free to change work processes (BUT team suggests improvements and conflict happen sometimes); Team has partial autonomy to pull tasks; Lack of discipline for TDD; Pair programming on demand
	Efficient coding	Culture of automation, creating tools.
Future	Assertive team	Little celebration on accomplishments; Improve communication to avoid conflicts; Giving sense of purpose and autonomy

The ambidexterity analysis has presented good results (null hypothesis rejected and positive percentages) for all aspects evaluated, with the exception of good alignment, for which the null hypothesis could not be rejected. Table 28 shows the data.

Table 28 - Ambidexterity data for Company E – Team 2

	Performance	Good Alignment	Bad Alignment	Adaptability	Bridging Ties	Strong Ties
Disagree	0.0%	0.0%	90.0%	0.0%	0.0%	0.0%
Neutral	5.0%	20.0%	10.0%	20.0%	13.3%	8.0%
Agree	95.0%	80.0%	0.0%	80.0%	86.7%	92.0%
p-value	0.000	0.135	0.005	0.002	0.000	0.000
Rejects H0	Yes	No	Yes	Yes	Yes	Yes

Projects success perception on this team shows that greater achievements were met with increased return on investment (the only evaluation greater than means of the other teams) and that (current and changing) customer demands are met, as shown in Table 29.

Table 29 - Projects success perception in Company E - Team 2

	Company C	Means for Other Teams	Std Dev
Reduced delivery schedules	3	4.4	0.9
Increased return on investment (ROI)	5	4.0	0.8
Increased ability to meet current customer requirements	4	4.0	0.5
Increased flexibility to meet changing customer requirements	4	4.3	1.2
Improved business processes	3	4.3	0.9

We can conclude, thus, that there is good ambidexterity perception on the team and projects success perception is mostly equals to or lower than the means of the other teams. Established agile practices and technical excellence focus summarize our perception of this team's current situation, in addition to the possibility of developing team assertiveness.

5.2.1.8 Company F

Company F is internationally recognized as an agile company which provides agile consultancy services and develops software on demand, with offices in several countries (3,000 employees in 12 countries and 30 offices). Most customers are in foreign countries and, for this reason, teams work remotely. Projects are developed with different customers and practices adopted to conduct the process are always agile, but with adaptations for each customer's context. In the project we analyzed, the customer is also agile and aligned with the company's values (i.e. technical excellence and continuous delivery). The team develops the front-end application integrated with existing Application Programming Interfaces (APIs), developed by other teams in the project, which are located in the United States of America.

The analysis of agile practices evolvement on this team was characterized by all practices already established in the past – since the setup of the office in Brazil – because of the influence of other offices around the world. Thus, this team is recognized as being notably experienced, because they run tests and deliver code in production automatically every each hour. Current and future focus is on sustaining outcomes with continuous improvement.

Figure 26 shows the outcomes they have already accomplished and continue sustaining, and Table 30 shows the evidence we found. Throughout the interviews with this team, as practitioners have wide experience in agile consulting, several comments were made on their ability to tailor the development process according to customer maturity. They even deal with customers that are not agile and wish to learn with them. In this case, they emphasized the importance of investing not only in managerial practices (i.e. Scrum), but also in technical excellence (i.e. XP practices and others) to allow continuous delivery.

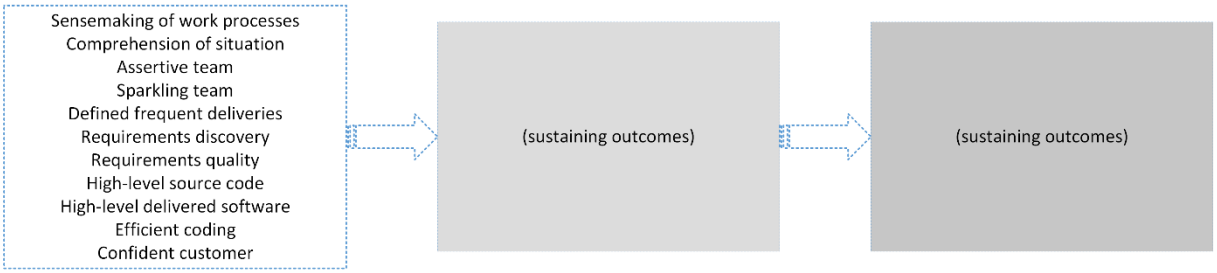


Figure 26 - Analysis of agile practices evolvement in Company F

Table 30 - Evidence for the outcomes identified for the team in Company F

Moment	Outcome	Evidence
Past	Sensemaking of work processes	Company is recognized internationally for being agile; processes are adapted according to customer context;
	Comprehension of situation	Measuring velocity (number of stories per week); when delays happen, team tries to understand why; flexibility of work processes (leaving estimates, pairing on demand…)
	Assertive team	Individuals are multifunctional (DEV + QA); tests status visible to everyone; developers participate in stories definition/breakdown/estimation; informally control WIP; working to remove impediments and improve all the time
	Sparkling team	Frequent job rotations; people are free to communicate and give new ideas; flexibility of work processes (estimates, pairing on demand)
	Defined frequent deliveries	Continuous flow of tasks; task is done when deployed in production; stories size is standardized; loose project deadlines; operations staff are linked to development staff
	Requirements discovery	Requirements are pre-defined before project starts; new requirements emerge during development (stories are divided/reprioritized); transparency with product owner; implementing fast spikes to support stories definition
	Requirements quality	Integration with other teams; intense communication with Product Owner; task is done when deployed in production; the team argues about requirements, the team participates actively in the creation of the requirements that actually add more value sooner
	High-level source code	Pair programming; test-driven development; code review (informal but mandatory)
	High-level delivered software	Code in repository is always healthy; critical bugs are rare and are solved quickly when they happen
	Efficient coding	Tools integrate communication and automation in work processes; continuous integration; collective ownership of source code; bugs solved quickly; automated tests (unit, integration and acceptance); automated deployment
	Confident customer	Task is done when deployed in production; task is accepted when PO has seen it (validates features and updates tasks status in the board); loose project deadlines (usually finishes before deadline, but delays happen sometimes, which is not a problem)
Present	(sustaining outcomes)	
Future	(sustaining outcomes)	

Only one null hypothesis for ambidexterity evaluation for this team was not rejected, the one for good alignment. Tests for the other null hypothesis resulted in rejection and with positive percentages. Table 31 shows these results.

Table 31 - Ambidexterity data for Company F

	Performance	Good Alignment	Bad Alignment	Adaptability	Bridging Ties	Strong Ties
Disagree	0.0%	0.0%	100.0%	0.0%	0.0%	0.0%
Neutral	12.9%	25.0%	0.0%	4.2%	4.2%	0.0%
Agree	87.1%	75.0%	0.0%	95.8%	95.8%	100.0%
p-value	0.000	0.056	0.000	0.000	0.000	0.000
Rejects H0	Yes	No	Yes	Yes	Yes	Yes

Projects success perception on this team is positive, with greater values than the means for other teams for all aspects we asked. Only the evaluation of increased return on investment remains similar to the means of other teams. Table 32 shows the values.

Table 32 - Projects success perception in Company F

	Company C	Means for Other Teams	Std Dev
Reduced delivery schedules	5	4.1	1.0
Increased return on investment (ROI)	4	4.1	0.8
Increased ability to meet current customer requirements	5	3.9	0.4
Increased flexibility to meet changing customer requirements	5	4.1	1.1
Improved business processes	5	4.0	0.9

In summary, the team in Company F has always been agile under the influence of other more experienced offices of the company around the world. Thus, agile practices have always been established and, currently, focus is on sustaining outcomes. Emphasis on technical excellence is also an aspect that we realized in this case. Projects success perception is positive, as well as ambidexterity agreement on the team.

5.2.1.9 Company G

This case shows a situation of agile practices evolvement opposite to the one presented in the team from Company F (Section 5.2.1.8). In this case, we evaluated a pilot project for agile adoption, in a big state-owned company. This company has accomplished CMMI Level 2 in the past (around eight years go) and is now going through a top-management initiative to adopt agile methods. In this project, the team was working on the implementation of a number of changes in an existing system, with an estimated deadline of four months.

This situation is shown in the agile evolvement analysis: few outcomes in the past, and a great deal of them in the present as well as planned outcomes for the future (Figure 27). Agile motion is an outcome placed in the past, as it represents the isolated initiatives for agile adoption in teams around the country. This agile motion evolved to an agile commitment when top-management members placed their focus on agile transformation, which is the current situation. During the pilot project, the team is accomplishing expected frequent deliveries, with planned sprints. Requirements are still gathered as the team lacks the experience to deal with user stories. High-level delivered software is a pursued outcome with emphasis on manual testing. The bureaucratic structure of the firm leads to the need of counting function points to achieve comprehension of the situation. The team is still responsive, with defined and separate roles. Regarding customer relationship, customer and team are still getting to know each other, which we characterize as “Team awareness of customer” and “Customer awareness of team”.

Figure 27 - Analysis of agile practices evolvement in Company G

Interviewees agree with the perception that, in the future, the company may focus on sensemaking of work processes, replacing Scrum "by the book" (which we called "agile learning") with practices adapted to their contexts. It is the perception that everything might be improved (see evidence on Table 33). The focus on creating a confident team and on enabling requirements to be discovered is also mentioned for future projects. The company is investing in creating an environment for automated tests and integration, as well as seeking efficient coding, which is evidence that agile priority is a pursued outcome. The evidence for the outcomes we identified is shown in Table 33.

Table 33 - Evidence for the outcomes identified for the team in Company G

Moment	Outcome	Evidence
Past	Agile motion	Isolated agile implementation initiatives; initiative to create an "agile process" (still prescriptive)
Present	Comprehension of situation	Count function points at the end of the sprint
	Agile learning	Pilot project; coaching; Difficulty to know how much is left to finish the task; product owner is still learning; communication in the team is very focused on the requirements analyst; implementing Scrum "by the book"
	Responsive team	Developers are still attached to their usual roles; team has free access to Product Owner but meetings are scheduled to talk to him; Problems with a complex story were not negotiated with Product Owner (story was not accepted); Scrum Master guides the team; the process tool drives decisions in the process
	Expected frequent deliveries	Sprint planning (3-week sprint, sprint backlog set to 30 points, burndown chart); could not deliver once because of a mistake by the product owner
	Requirements gathering	Update use cases during the sprint; difficulty to create small stories; "ready" concept still needs adjustments
	Requirements quality	Tests analyst helps requirements analyst defining requirements with quality
	High-level delivered software	Manual testing for each task; tester tests beyond the acceptance criteria of the task; regression test at the end of the sprint; having environments to test and validate; daily commit; publication is semi-automatic
	Team awareness of customer	With agile methods developers and testers get to know about the business; team defines stories with Product Owner
	Customer awareness of team	Requirements analyst helps Product Owner in his activities; Product Owner is participative and collaborative; transparency with customer to implement spikes with test automation

	Agile commitment	Strong top management support and coaching; deployment is bureaucratic (accesses are limited); team was changed during the project; people in organization still work in more than one project at the same time; Product Owner was assigned by top management and did not have knowledge about the system; need more training and more coaching
Future	Sensemaking of work processes	Feeling that everything might be improved
	Confident team	Trying to stimulate multiple roles in developers; Focus on creating a "team value"
	Requirements discovery	Sprint planning, validation is performed for each story, but difficulty in accommodating changes in requirements and stories size still vary a great extent
	Efficient coding	Continuous integration environment not ready yet; proof of concept to implement automated tests
	Agile priority	Company is developing an agile roadmap; focus on changing the organizational culture; starting investments in devops; starting investments in automation (integration, test and deployment); more people needed for working on automation; more people need for working with agile projects

Ambidexterity perception data reflect the context of the company, showing the team still lacks agreement on bad alignment aspects (null hypothesis could not be rejected). For all the others, the Chi-Square tests rejected the null hypothesis (showing agreement within the team). Data show percentages for adaptability evaluation that reflect that the team does not agree on adaptability capabilities of the company, and 75% of staff preferred not to express their opinion on the good alignment of management systems. Table 34 shows the data.

Table 34 - Ambidexterity data for Company G

	Performance	Good Alignment	Bad Alignment	Adaptability	Bridging Ties	Strong Ties
Disagree	0.0%	12.5%	37.5%	66.7%	0.0%	0.0%
Neutral	9.4%	75.0%	31.3%	33.3%	12.5%	0.0%
Agree	90.6%	12.5%	31.3%	0.0%	87.5%	100.0%
p-value	0.000	0.001	0.511	0.000	0.000	0.000
Rejects H0	Yes	Yes	No	Yes	Yes	Yes

Project success perception on this team is positive, and that was mentioned during the interviews. Our perception is that, in comparison with the processes that the team adopted before the pilot project, the agile processes brought great benefits. Quantitative data show this positive evaluation: similar or greater evaluation than other teams' with regard to all aspects, as shown in Table 35.

Table 35 - Project success perception in Company G

	Company C	Means of Other Teams	Std Dev
Reduced delivery schedules	5	4.1	1.0
Increased return on investment (ROI)	4	4.1	0.8
Increased ability to meet current customer requirements	5	4.0	0.5
Increased flexibility to meet changing customer requirements	5	4.1	1.1
Improved business processes	5	4.0	0.9

During the validation meeting in this company, which happened two months after the end of this project, the Scrum Master and the functional manager commented that agile adoption in the company was still very incipient. The feeling was that top-management decided to adopt agile methods based on external stimuli, rather than based on the initiatives of the “agile motion” we identified in the research. They believe it frustrates people that were – even isolated – working with agile methods because the knowledge within the company was ignored. They also emphasized that the outcome “agile priority” is a wishful thinking. It is something that they do not really feel as something that is going to happen in near future.

The case presented for Company G shows a case of a team that is starting with agile methods, still learning the practices. The context of a big and bureaucratic company imposes some challenges and shows the importance of top-management support. Ambidexterity data reflect lack of alignment and adaptability, although project success perception is positive.

The individual results we presented here in each of the within-case analyses were also presented to the team leaders, as feedback of the research and to evaluate our interpretation. The suggestions they gave were considered in the presentation of the data. Next section shows the findings of the cross-case analysis.

5.2.2 Cross-case analysis

The cross-case analysis forces the researcher to go beyond individual results and search for patterns in the data (EISENHARDT, 1989). We performed this cross-case analysis searching for evidence in the data to confirm or refute the cases study propositions (Figure 11 on page 54).

Our first proposition stated that *the team plays a central role in agile software development maturity*. To evaluate this proposition, we analyzed the team evolvement in the maturing processes of the cases. The main evidence for this proposition are the cases of Companies C and G, which still had responsive teams and lacked ambidexterity perception on their teams (Table 36). It means that both teams which still did not feel that they had adaptability to work (essential to enable agility) were also experiencing difficulties to become confident. Conversely, one of the best results in ambidexterity, Company F, was the only one identified as having a sparkling team.

Secondary evidence to analyze this proposition is that, while our definition for maturity in agile software development includes standardization and continuous improvement of agile practices, our teams have matured over time by increasing assertiveness and motivation to learn. We believe this relation shows team's conduct in central to the evolvement of agile work processes.

Accordingly, we lack enough data to claim that thorough definition and standardization of processes (focus on exploitation) grow as agile practices evolve. Instead, teams apply some practices to comprehend their situation (e.g. project tracking, report work status, simple metrics etc.) but their evolvement is perceived in their initiative to improve continuously. We conclude, then, that we have enough evidence to confirm our first proposition: the team plays this central role and we emphasize that it is supported by an ambidextrous management, as shown in the next proposition.

Our second proposition speculated that *teams get mature in agile software development by combining exploration and exploitation activities, that is, through ambidexterity*. This proposition can be verified in our quantitative data. Considering that maturity should be related to the success of the projects (MAIER; MOUTRIE; CLARKSON, 2012; LEPPÄNEN, 2013), we show in Table 36 the means for project success perception on each team and the ambidexterity data for each case. We present in this table the percentage of agreement in each team for each aspect.

Table 36 - Ambidexterity data from all cases

Team	Projects success perception	Null hypothesis not rejected	Good performance on team	Good alignment*	Bad alignment*	Adaptability*	Bridging ties	Strong ties
Company A	4.2	0	87.5%	100.0%	6.3%	91.7%	100.0%	95.0%
Company B Team 1	4.6	2	80.0%	40.0%	80.0%	20.0%	100.0%	100.0%
Company B Team 2	3.8	3	67.9%	57.1%	57.1%	61.9%	100.0%	94.3%
Company C	3.4	3	58.3%	16.7%	41.7%	27.8%	88.9%	90.0%
Company D	4.4	2	87.5%	75.0%	12.5%	91.7%	83.3%	100.0%
Company E Team 1	3.6	0	92.5%	90.0%	5.0%	50.0%	83.3%	92.0%
Company E Team 2	3.8	1	95.0%	80.0%	0.0%	80.0%	86.7%	92.0%
Company F	4.8	1	87.1%	75.0%	0.0%	95.8%	95.8%	100.0%
Company G	4.6	1	90.6%	12.5%	31.3%	0.0%	87.5%	100.0%

* Underlined data show null hypotheses which could not be rejected in Chi-Square tests; thus, they do not represent an agreement in the team.

Ambidexterity evaluation based on bridging ties and strong ties confirmed that all agile teams successfully combine these characteristics. Regarding alignment and adaptability, the teams with a greater number of non-rejected null hypotheses (i.e., less agreement on the team) have the lowest perception of good performance on the team and projects success perception mean lower than 4. The teams with none of the null hypothesis rejected, presented high perception of good performance on the team (close to 90%), high percentages of good alignment; and low percentages of bad alignment. Teams with just one non-rejected null hypothesis present a similar situation. With respect to adaptability, the teams with none or one non-rejected null hypothesis also feel adaptability in management systems (with one exception, for team 1 in Company E, which presents a lower percentage). We have not considered the team from Company G in this evaluation (note the extremely low values for alignment and adaptability) because it represents a pilot project; thus, the perception of the team considers a recent and short experience.

In addition to that, management appeared in the cases and plays an essential role in the evolvement of the team. Responsive teams can evolve to confident teams when they find space for autonomy. Ambidextrous management is the one that allows this autonomy by keeping team alignment with adaptability. A sparkling team could only be identified in Company F, one of the best results in ambidexterity. With these data, we consider that the second proposition was confirmed.

The conclusion for the third proposition is subjective, yet evident. This proposition stated that *the exact set of practices is not pre-defined at each maturing stage*. It has been confirmed by the lack of pattern among the contexts and practices on each team. They started their agile adoption differently and the practices also evolved differently. We could not identify maturity stages but we could, however, uncover that the outcomes are similar, and a pattern could be identified in the progression of these outcomes, as described next in the evaluation of the fourth proposition.

The fourth proposition, which posed that *teams evolve in agile development starting with agile values, involved customer, planning and requirements; and then, later invest in agile coding and agile testing*, was <u>not</u> confirmed. As practices and contexts are too different among teams, we could not identify this pattern of adoption on the teams included in this study: there was not such a sequence of practices adopted.

However, as a pattern in the outcomes emerged from the comparison of the cases, our cross-case analysis uncovered seven categories of pursued outcomes – or evolvement threads – and how they evolved on actual agile teams, as shown in Figure 17 at the beginning of Section 5.2. The categories are: practices learning, team conduct, deliveries pace, features disclosure, software product, customer relationship and organizational support. We represented them in the Progressive Outcomes framework for agile software development (Figure 17). Table 9, Table 12, Table 15, Table 18, Table 21, Table 24, Table 27, Table 30 and Table 33 show the variety of practices the teams implemented to pursue the outcomes described in this section. One could argue that not all the outcomes identified in the cases are included in this framework. Indeed, the outcomes Standardization of agile practices, Specialist team, Interdisciplinary team, Represented Customer, Third-people supportable software were specific to the context and the moment a specific team was experiencing and, for this reason, were not included in the general framework.

This framework represents a non-linear and dynamic, i.e. discontinuous, process of agile software development evolvement. In the validation presentation that we made with the team leaders, we presented which of the outcomes each team had accomplished (in a subjective assessment, presented in APPENDIX H), and it was clear that each team evolves in the outcomes that are relevant to their business contexts. For example, Company A did not invest in “Agile learning”, as they hired an experienced team, and they did not implement initiatives to allow for requirements discovery, either. In their business context, the requirements usually did not change once they had been defined. Company C had “Comprehension of situation” before agile adoption. Company B – Team 1 did not pursue any of the customers’ outcomes, as they did not have this need. Company F did not accomplish the outcomes of “responsive” and “confident team” before the “assertive team”, as it has an established culture based on other mature offices around the world.

We did not identify stages for agile adoption, neither a predefined sequence for pursuing the outcomes. Furthermore, we consider that these outcomes are highly volatile and any small change in an aspect may lead the team to take a step behind. For example, when a customer changes, outcomes need to be pursued again. When a team changes, the same takes place. If technology changes, an efficient coding might also be lost. Thus, any tentative assessment based on this framework will be a

temporary picture of the dynamic situation teams face in agile work processes improvement.

5.3 Agile Compass: the diagnosis tool

In Section 4.2.3, we described the pre-evaluation we performed with agile practitioners. One of the main types of feedback we received was that the Progressive Outcomes Framework successfully described the mechanisms agile teams apply to mature, but it lacked clues for teams to identify where they are situated in this maturing process (this and other feedback information is described in Section 5.4).

For this reason, we complemented our main result – the Progressive Outcomes framework – with a diagnosis tool to allow the self-assessment of agile teams. We name it as the Agile Compass, as the objective is to provide a simple, slack and temporary picture of the outcomes a team has accomplished. To create it, we consolidated all the evidences across teams for each outcome. The Agile Compass was published in Fontana et al. (2015d).

Each outcome was thus characterized with a short sentence and with two or three items in a checklist. Teams should, in a collective meeting (such as in a retrospective meeting), identify which items they observe as applicable to their reality. If all items in an outcome checklist are marked, we consider the outcome was achieved. We suggest that the Progressive Outcomes framework should be printed and visually available, with accomplished and desired outcomes highlighted, to stimulate constant self-assessment and self-improvement. By being visible to the team, the framework may provide information for software engineers to reflect and address the need of tools that “encourage sensemaking, critiquing, and the identification of new forms of development work” (DYBÅ; MAIDEN; GLASS, 2014, p. 32).

Recent research has shown that the stage-based models for agile maturity, as we presented in CHAPTER 2, have not been successfully validated and even criticized for not embracing the agility characteristics in the software process (GANDOMANI; NAFCHI, 2015; GREN; TORKAR; FELDT, 2015). By agreeing with these authors, we built the Agile Compass based on empirical data, i.e., reflecting actual agile teams dynamics. Our proposal addresses, thus, the need for allowing adaptability with clear guidance: teams identify where they are by diagnosing accomplished outcomes and discuss their own ways to improve or pursue new outcomes.

Table 37 shows the categories, their outcomes and the statements that should be used to identify whether outcomes have been accomplished.

Table 37 - The Agile Compass: a diagnosis tool to identify accomplished outcomes

PRACTICES LEARNING	**Agile trial** The team is trying to use an agile method with empirical learning. Most of the times, not all practices were implemented. Benefits are not fully realized. ☐ We are learning on the fly ☐ We did not realize the expected benefits of agile adoption ☐ We are still learning how to deliver value to our customer **Agile learning** The team is learning the agile method. The method is used "by the book" usually by getting training or coaching. Practices are fully implemented, as described in books and tutorials. It is important for the team to learn the value of each practice to be able to tailor or even abandon them later. ☐ We are following an agile method "by the book" ☐ We are getting training or coaching ☐ We are working with pilot projects **Sensemaking of work processes** The team understands agile practices and the value they add. They can now tailor the practices to their context and feel confident to be flexible in work processes, without abandoning agility. ☐ We learned the agile method and understand the value of each practice ☐ We are currently tailoring the practices ☐ We feel the work processes are more flexible according to the characteristics of each project **Comprehension of situation** The team has information about their work processes. Information is simple and usually based on physical visibility or simple metrics. They use this information to make decisions and implement improvements in the process. These improvements are usually small and incremental. ☐ We are able to explain what happens in our work processes ☐ We have some metrics ☐ We use our understanding to make decisions
TEAM CONDUCT	**Responsive team** The team needs close leadership to drive activities. They do not have/were given autonomy to make decisions. The team does not have the confidence to protect their work, that is, overtime is usual because of extra demands. ☐ We have partial autonomy to make decisions (about design, technology, project etc.) ☐ We feel the need to have the Scrum Master with us most of the time ☐ We have weak politics to protect team work **Confident team** This team still needs the Scrum Master around, but feels confident to make little decisions mainly about the project. Their communication is fluent. Changes in work processes need to be approved by management. ☐ We are fluent with practices and feel autonomous to make some decisions in the project ☐ We know each other and are free to communicate ☐ We suggest improvements in the process, but they usually need approval **Assertive team** This team feels responsible for the project. They also have the autonomy to change the process. They make their own decisions and inform management. ☐ We have the autonomy to improve our work processes ☐ We have the capability to drive our own work during the project ☐ We have politics to protect our work **Sparkling team** This team is focused on technical excellence. They create their work processes, supported by management. They need a supportive environment to continue learning and, thus, keep motivated. It is possible to feel the focus on constant improvement. ☐ We define our work processes and change them whenever needed ☐ We feel highly motivated to continue learning ☐ We feel leadership supporting the implementation of our ideas
DELIVERIES PACE	**Expected frequent finished coding** Sprints are planned to finish code and integrate it into the repository. Most of times, functionality is not tested yet. The team is learning how to deliver value at the end of the sprint. ☐ At the end of the sprint, we have code finished, but we do not deliver it ☐ Our sprint size sometimes varies ☐ We are not sure of what adds value to our customer **Expected frequent deliverables** This team identifies the value to be delivered but cannot deliver at the end of the sprint. The environment is not fully supportive of agile build and code integration. Testing is usually manual.

☐ We are capable of having functionalities ready to deliver in the end of the sprint, but we do not deliver
☐ We can identify a minimum releasable product

Expected frequent deliveries

Deliveries are planned and done, but usually late. Sometimes extra demands appear and the team has to address them. Environment is more supportive for automated build and integration

☐ We plan our deliveries and do them, usually late
☐ Our stories sizes vary

Defined frequent deliveries

Deliveries are planned and done within the deadline, sometimes before. Stories are usually small. Environment supports automated build and integration.

☐ We plan our deliveries and do them, sometimes before deadline.
☐ Our stories sizes are similar, usually small

FEATURES DISCLOSURE

Requirements gathering

Requirements are defined at the beginning of the project, usually with texts or diagrams. The team does not feel comfortable about changes or emerging requirements during the project.

☐ We define most of the requirements at the beginning of the project
☐ We do not exactly know how to deal with changes in requirements during the project

Requirements discovery

Requirements are vaguely defined at the beginning of the project; they are detailed as sprints are initiated. Changes and emergent requirements are welcome. Spykes might be used for requirements elicitation.

☐ We define, in detail, the requirements for the next sprint
☐ We are comfortable about changes or emergent requirements
☐ We can identify a minimum releasable product in our stories

Requirements quality

There are clear initiatives to guarantee the quality of the requirements, for them to be in accordance to customer needs.

☐ We are concerned with whether what we deliver to our customer is exactly what the customer needs
☐ We endeavor initiatives to improve the quality of the requirements

SOFTWARE PRODUCT

High-level source code

Coding is an important activity and there are initiatives to guarantee the code is clear and robust, using best practices available.

☐ We care about our code
☐ We endeavor initiatives to guarantee our code is fine, such as reviews, refactoring, pair programming or others

Awareness of failures

There are failures in deliveries that need to be treated as soon as possible, and failures happen because of problems in development process

☐ We spend some time in the sprint correcting bugs from the previous sprint
☐ We spend some time in the sprint with issues in code building and integration

High-level delivered software

There is a concern with the quality of the feature being delivered. For this reason, testing is a priority, either automated or not. Also, there is a concern with the integrity of the code in the repository.

☐ We see testing as a priority
☐ Code is integrated as soon as possible
☐ Our code repository is always healthy

Efficient coding

Coding, integration and testing are performed with an infrastructure that allows agility. The team is concerned with removing unnecessary delays in work processes. Devops is a common infrastructure to allow efficient coding.

☐ We have automated tests, either unit or functional tests
☐ Our build and integration are automated
☐ We are concerned with removing delays in work processes by automating manual tasks

CUSTOMER RELATIONSHIP

Team awareness of customer

The team is learning how the customer business is and the dynamics of the customer demands. This is the reason why their contribution to customer requirements is still incipient.

☐ We are getting to know our customer and his/her demands
☐ We have partial ability to help customer defining requirements

Customer awareness of team

It is the learning process of the customer about the team. The customer is getting to know the processes, but is not fully used to that. Sometimes it seems there is a lack of trust on the team.

☐ Our customer is learning how we work
☐ The customer knows what we are going to deliver
☐ The customer is not fully comfortable about reprioritizing requirements

	Confident customer The customer is confident about the team's work. He believes that the team is concerned about delivering what he needs. The customer is usually more flexible about deliveries and is more comfortable about reprioritizing requirements. There is transparency between the team and the customer. ☐ We assist customer in requirements definition ☐ Customer is aware of his role in the process ☐ We feel the customer trusts our work **Partner customer** The customer recognizes the partnership with the team: while he feels the team is committed with his business, the team feels the customer as being part of the team. ☐ We identify and suggest improvements to our customer's business ☐ Our customer feels as if he is part of our team, with shared responsibility about deliveries
ORGANIZATIONAL SUPPORT	
	Agile motion Some isolated teams are starting to work with agile. The company is aware but not concerned about acknowledging the process or the results. ☐ We see isolated agile initiatives in our company ☐ We feel weak acknowledgment of top management about these initiatives **Agile commitment** Top management decided to implement agile in the company and initiatives are official. There are investments with training, coaching, communication and infrastructure for agile transformation ☐ There are official agile pilot projects in the company ☐ We feel company support (training, coaching or infrastructure) for agile transformation **Agile priority** There is full support from top management to agile transformation. Departments, roles, teams change to support agility. As it is a top-bottom initiative, resistance still appears on some teams. ☐ We see that the company' structure (physical and departmental) has changed to support agile transformation ☐ We feel there is some resistance to change in some teams **Agile business** Software development companies that are created with agile methods are, usually, agile businesses. The company management strategies are focused on people and leanness, not just on software development, but on the whole company. ☐ Our whole company is recognized as being agile ☐ Our teams adopt an agile foundation, but customize the way they work according to the characteristics of projects

5.4 Results Evaluation

The results of this thesis were evaluated at two different moments. The first was the pre-evaluation based on preliminary results, through a survey with agile practitioners, described in Section 4.2.3. The second was the evaluation of final results performed with individual interviews with agile researchers and practitioners, as described in Section 4.3.

In the pre-evaluation phase, we obtained feedback about the Progressive Outcomes framework from 231 agile practitioners. The results of this survey was published in Fontana et al. (2015c). The profile of the respondents is presented in Table 38. It shows the percentage of respondents for each range of experience in software engineering in general and particularly in agile methods.

Table 38 - Respondents' profile in the pre-evaluation

Experience in software engineering (years)	
0 to 3 years	26%
4 to 10 years	27%
> 10 years	37%
Not informed	115
Experience in agile software development (years)	

0 to 1 year	40%
2 to 4 years	38%
> 4 years	19%
Not informed	3%

Considering the perception of all respondents, Table 39 shows the percentage of agreement, neutral opinion and disagreement with each of the statements evaluated by respondents. As the perception of experienced practitioners is relevant when evaluating maturity, Table 40 filters responses with the opinion of the ninety-three practitioners with three or more years of experience in agile. Highest percentages are highlighted in boldface.

Table 39 - Opinion of all practitioners about the Progressive Outcomes Framework (pre-evaluation)

The Progressive Outcomes framework…	Opinion	Percentage
…is useful to aid teams to evolve with agile methods	Disagree	3%
	No opinion	4%
	Agree	**94%**
…is useful to define what maturing is in agile	Disagree	1%
	No opinion	3%
	Agree	**95%**
… is easy to understand	Disagree	5%
	No opinion	6%
	Agree	**88%**
… comprises what I believe is required to evolve in agile	Disagree	6%
	No opinion	11%
	Agree	**84%**
… includes unnecessary information	**Disagree**	**56%**
	No opinion	29%
	Agree	15%
… allows me to identify the current situation on my team/company	Disagree	8%
	No opinion	7%
	Agree	**85%**
… is adaptable to different organizational contexts	Disagree	5%
	No opinion	17%
	Agree	**77%**

Table 40 - Opinion of experienced practitioners about the Progressive Outcomes Framework (pre-evaluation)

The Progressive Outcomes framework…	Opinion	Percentage
… is useful to aid teams to evolve with agile methods	Disagree	4%
	No opinion	3%
	Agree	**92%**
… is useful to define what maturing is in agile	Disagree	1%
	No opinion	3%
	Agree	**96%**
… is easy to understand	Disagree	5%
	No opinion	5%
	Agree	**89%**
… comprises what I believe is required to evolve in agile	Disagree	12%
	No opinion	8%
	Agree	**80%**
… includes unnecessary information	**Disagree**	**58%**
	No opinion	26%
	Agree	16%
… allows me to identify the current situation on my team/company	Disagree	16%
	No opinion	2%
	Agree	**82%**
… is adaptable to different organizational contexts	Disagree	9%
	No opinion	17%
	Agree	**74%**

Percentages of responses with agreement, neutral opinion and disagreement show accordance when comparing all responses with responses of experienced practitioners only. The majority of respondents agreed that the preliminary version of the Progressive Outcomes framework was useful to aid teams to evolve with agile methods, comprised what they believed was required to evolve in agility, allowed them to identify their current situation and seemed to be adaptable to different organizational contexts. There was least agreement on the statement which evaluated if the framework included unnecessary information. Even though, nearly half of practitioners disagreed.

In one of the events where we collected data, the questionnaire also had an open-ended question asking for comments or suggestions. The consolidation of these responses and feedback received after our talks showed that 1) practitioners felt that the evolution of organizational context should have been represented in the framework[5]; 2) more evidence was necessary for a team to identify whether an outcome was accomplished; and 3) they lacked a clear description of management initiatives in the maturing process.

Suggestions were included in the second round of case studies: we added an analysis of organizational position in agile adoption, for the teams in our sample that were within big companies. We also built a diagnosis tool to provide clues based on the evidence on how the teams in this study accomplished the outcomes. Finally, we included, in the discussion of results, the evidence we found from management in mature teams.

The second evaluation was performed over the final results, with in-depth presentations and interviews with agile practitioners and researchers. Our first round of interviews took place during the Agile Trends 2015 conference in São Paulo. We performed four presentations followed by semi-structured questions to collect the feedback of interviewees. The conversations took forty minutes on average and the feedback we received is summarized in Table 41. The interviewees' comments refer to the previous version of the framework in which categories were respectively called: practices, team, deliveries, requirements, product, customer and organization.

[5] The Progressive Outcomes framework version presented in the pre-evaluation was based in the four first case studies and, thus, did not include the analysis of the organizational support evolvement.

Table 41 - Feedback received in the first round of evaluations – Agile Trends 2015

Respondent 1 – Director of Product Safety – 17 years of experience in agile methods	
Utility	He agreed on the utility of the framework to situate agile teams, and also agreed on the dynamics we represent. He commented he would call our result a "competence" – instead of "maturity" – framework. In his company, they use a similar tool to evaluate customers' readiness to adopt agile methods. It helps agile coaches to understand which value could be delivered to which customer. He also mentioned our framework represents the complexity of agile teams and would call it a tool for "comprehension" and not "evaluation". To be applicable in practice, the framework should give hints on the "sensors" or "metrics" to identify the outcomes. Regarding certification of agile teams, he said it is possible, but he would not recommend it.
Quality	He would add two outcomes in the "deliveries" category. One before the ones we proposed, when teams cannot even finish code. Another after the ones we proposed, which he called "Scaled delivery", when teams implement continuous deployment. In the "requirements" category, he included the idea of defining hypothesis, instead of requirements, to get to "requirements quality". He made additional comments with respect to naming in categories: the product category could be called "technique"; "product" category is more related to "requirements", in his opinion, and the term "practices" seems to be confusing. Regarding the figure of the framework, he commented that it does not reflect the complexity of agile evolvement, and suggested a format that would not relate outcomes to a sequence ("more art and less power point").
Efficacy	For him, our results seem to be closer to reality than other models, such as the Agile Fluency[6]. He understood management ambidexterity and recognized it is a concept that makes clear that agile teams, although self-organized, need guidance. He agreed the framework seems to be applicable to different contexts.
Locality	He has experience with teams worldwide and recognizes the framework is applicable in other countries.
Respondent 2 – Scrum Master – 7 years of experience in agile methods	
Utility	His opinion is that the framework addresses a government need to choose software suppliers. As government in Brazil is widely adopting agile methods, our results would be useful to aid in the assessment of a company to supply software to government projects. He emphasized it would not be done with a certification – certification is not applicable to agility – it would not be a ranking, but an assessment of the current situation, to meet a specific need. He also saw utility for the self-evaluation of the team. A Scrum Master could use it to show teams where they are, how they evolved, where they could go, as an essential movement towards continuous improvement. He also recognized our results as providing a basis for the awareness of the transformations in organizational structures and leadership styles. He sees that future studies could use our framework as a basis to deepen the organizational mechanisms for agile transformation.
Quality	He commented that maybe we could make improvements on a few terms. In his company, for example, they do not work with requirements, but with hypotheses, and it does not appear clearly in the "requirements" category.
Efficacy	For him, the framework represents the complexity of an agile team, mainly when we make clear that is a temporary picture, such as the financial situation of a company. He understands that our result is a set of principles, and not a model. These principles might be followed with different models (which define the shape). Thus, he considers that our purpose fits different organizational contexts.
Locality	He commented he has experience in South America and, in such context, the framework seems to be applicable.
Respondent 3 - Agile Coach and Trainer – 9 years of experience in agile methods	
Utility	He found it interesting that we have formalized a concept of maturity specific to agile methods, in opposition to current established reference models. He also added that the framework would allow each individual (from team members to managers) to look at the "map" and identify how one sees a team's outcomes. It would be excellent for an agile coach to understand the situation of the company, and identify the outcomes where work should be focused on. He commented the framework is big for a team to be able to self-evaluate, it is a job that could be done with a coach, using a team exercise. For him, the sequence in which we placed the outcomes may aid teams to identify the next steps. Regarding certification, he commented that a maturity tag would be interesting, but he thinks it is impossible, given the complexity of the dynamic of agile teams. He also sees that an interesting future study could identify the paths that different companies – in their specific contexts – take in agile evolvement.
Quality	He commented that the framework does not clearly show software product metrics. For him, these metrics are input to requirements discovery. He also commented that the term "requirements" should be changed. Based on his experience, in agile methods, people do not define requirements, but hypotheses. However, he also added that it is hard to change, because people in the field are used to this term. He mentioned that naming the "practices" may be confusing with respect to what they mean and also suggested that we could make clarify the continuous improvement dynamics in the framework. He said that a low-maturity team would not continuously improve their own work, because it is a characteristic of more mature teams. He thinks it could somehow appear in the framework.
Efficacy	He agreed that our results effectively represent what and how maturity is in agile software development. He also commented that he agrees that the framework suits different contexts.
Locality	Based on his experience, the framework seems to be applicable to contexts in other countries, even different cultures, e.g. India.
Respondent 4 – Agile Coach – 4 years of experience in agile methods; and Respondent 5 – Researcher – 2 years of experience in agile methods	
Utility	Respondent 4 commented on the utility for the company – considering its own context (restrictions and wishes) – to draw a target of where to go in agile transformation and experiment a diversity of practices. Then, the company could identify whether the outcomes were accomplished. Besides, he believes that our results would make agility and its adoption tangible for inexperienced teams. For him, the framework brings to light the dimensions involved in agile transformation. Respondent 4 and Respondent 5 agreed on the utility of the framework to evaluate a team. He said it is great to give feedback to the team with respect to where they stand. It allows them to know what the next step is, and what the team needs to achieve it. He commented he tried to create a similar tool for evaluation of teams. Regarding certification, Respondent 4 mentioned that it

[6] Agile Fluency is a four-stage model for agile adoption, proposed by Diana Larsen and James Shore: http://martinfowler.com/articles/agileFluency.html

	would be bad: it leads to stagnation; he also mentioned the commercial purpose of selling certificates and the search people do for the tag rather than for the improvement itself. Respondent 5 saw the framework as the beginning of a theory for evolvement in agile transformation.
Quality	Respondent 4 commented that our framework requires experience in knowing the meaning of each outcome to be used. Inexperienced individuals would not be able to understand it. Respondent 4 commented that, at first sight, the "Practices" category was a little confusing because practices should be spread all over the other outcomes. Besides, the "requirements" category does not make it clear what the difference is between "quality" and "discovery". Still as regards understanding of outcomes, Respondent 5 commented that the teams' category lacked "high performance". The layout of the outcomes gave Respondent 4 and Respondent 5 a sense of sequence, of levels in accomplishing the outcomes. Respondent 5 understood that it is a sequence, but not necessarily a linear one.
Efficacy	Respondent 4 commented that the framework is adaptable to different contexts. Teams of different companies could see the map and identify where they are and how to get to outcomes in their contexts.
Locality	Not evaluated.

The second round of interviews was performed during the XP 2015 conference, in Helsinki. By applying the same method we used in the first round, we performed six presentations, followed by an interview with semi-structured questions. Each discussion took one hour on average, and their perceptions are summarized in Table 42. Similarly to the comments in Table 41, the interviewees' comments from XP 2015 refer to the previous version of the framework in which categories were respectively called: practices, team, deliveries, requirements, product, customer and organization.

Table 42 - Feedbacks received in the second round of evaluation – XP 2015

Respondent 6 – Norway – Researcher – 3 years of experience in agile methods Respondent 7 – Brazil – Researcher – no experience in agile methods Respondent 8 – Brazil – Functional Test Lead – 2 years of experience in agile methods	
Utility	Respondent 6 finds the results help systematize some ad-hoc perceptions researchers have when studying agile teams. She told a story about a company in Europe where she and her colleagues went for evaluation of the process. She identified, in this case, all of the categories we presented in the framework. However, at the time of their study, they did not have a systematic procedure to organize their observations. The framework could help on that. Respondent 6 also added the framework is a useful tool for consultants to make an initial assessment of the company. They could identify categories where attention should be focused on for agile improvement. Respondent 7 agreed with her opinion, saying that the framework brings "awareness" of what will be faced in agile evolvement. Respondent 7 also added that the framework could help teams to identify their current situation. From the research point of view, they commented that our results do not deepen the analysis of each category. Thus, the framework provides room for future research. For each category, theories from other fields and more empirical data could be collected. Some questions that remained open were about the utility of the framework to different roles, such as project managers, Scrum Masters, low-maturity teams, high-maturity teams, developers, etc. In addition, the questions we asked led to a discussion on how the result could be used by a company that is about to start agile transformation.
Quality	According to Respondent 6, the name of the thesis, "maturity in agile software development", does not attract people's attention. The software community in Europe does not appreciate maturity models. Certifications, for example, are not their interest. Respondent 7 suggested it is an "evolvement" model, instead of a "maturity" model. Regarding the content of the framework, they said they wished they had clearly seen in our outcomes: autonomy, individual knowledge, commitment and self-managing in the team category; quality and tests issues; measurements and definition of done. Sabrina also added that the framework could also have provided more clues on which outcomes should be pursued. Regarding the completeness of the model, they suggested that the influence of practices in outcomes could more deeply analyzed.
Efficacy	Respondent 6 identified in the framework one of the teams she studied in Poland. Respondent 8 could also map a team she worked with in Brazil. Respondent 6 and Respondent 7 commented they appreciate our proposal for fast and small cycles of improvement. Respondent 7 added that she is working on an agile transformation in a company in the south of Brazil and they were looking for a step-by-step to implement agile. They arrived to the same conclusion as we did in our study: there is no such a thing as a unique path of implementation.
Locality	Respondent 6 identified the team she studied in Poland, evidence that maybe the framework is applicable in other countries. Their main comment with regard to locality was that the importance of the framework as a support for agile transformation is more intense in Brazil, as it is a country that is beginning agile adoption. Europe has another reality and, for consultants there, maybe the framework does not add anything new. Besides, maturity certification in Europe is not a topic of common interest.
Respondent 9 – China – Agile Coach – 5 years of experience in agile methods	
Utility	His first comment was on the possibility of using the framework with a team to evaluate their situation. He commented that in his company they have a real need. They have no idea of the maturity level of their 41 Scrum teams. He would like to have a quick scan to understand how many teams are immature, mid-level or mature.
Quality	He said the framework has observable results, helps see the gaps but does not show how to improve. "Seeing is ok, but so what?", he said. He understands it is a challenge, because a prescription for one team may not apply to another. But managers always ask: "can you help me move to the next level?" He said that consultants cannot tell managers they have to experiment

	endlessly. Regarding the understanding of the dynamics, he agrees with the temporary picture the framework provides to maturity. He had difficulty in understanding the difference between delivery and product. He said he wished the framework had addressed team culture and team dynamics more clearly. He also suggested we should provide links between the outcomes, showing the dependences among them.
Efficacy	He identified one of the teams he was working with, confirming the suitability of the framework to different contexts.
Locality	He told us that the framework seems to be suitable to a Chinese team he evaluated.
Respondent 10 – Denmark – Agile Coach – 15 years of experience in agile methods	
Utility	He thinks that the framework is useful for teams to find out where they stand. For the team, the value lies in the discussion that could be fostered by a diagnosis using the framework. His opinion is that discussion is more important than the framework people are discussing over. Maybe by using our framework the team would think and discuss about things they would not otherwise.
Quality	In general, he said the framework reminds him of TPI - Test Process Improvement[7] and also tried to compare team category with Tuckman's stages of team evolvement (TUCKMAN, 1965). He likes the idea of not having a ranking and the outcomes made sense to him. He believes that using the framework in practice would lead to discovering other outcomes. He also added a comment on the ambidexterity concept. He said that he identifies it in a context where you make it more flexible sometimes by constraining a situation. Continuous integration, for him, is a good example of defining rigid rules on how the code is handled and released. However, it leads to more flexibility and doing things faster. “The magic lies in the balance”, he said.
Efficacy	He thinks the framework shows good empirical results and one good way to improve it is by starting to use it. He thinks one organization could add a few outcomes, another could add different ones. He thinks that maybe there might be differences in the product category if the team work with different products, such as UX (User Experience) software.
Locality	He said that the framework should be open in order to be tailored to each organizational context.
Respondent 11 – Norway – Researcher – 9 years of experience in agile methods	
Utility	He kindly asked to refer to our conversation as a discussion on the research work, instead of an evaluation. He found value in our results as empirical work that would be useful to compare with similar work in other countries. On the other hand, he considered the framework as huge. He suggested we should consider, in future work, focusing on in each dimension as a research topic. More mature research fields provide theories and evidence that could be compared to our result.
Quality	His opinion is that the framework is quite broad. Each of the categories could be treated as a whole research study. For example, in the team category, there are plenty of theories that explain team evolvement. Also for organizational evolvement, he mentioned the Enterprise Agile studies. He added that, for a large-scale scope, it seems that some extra challenges could be added to the framework.
Efficacy	He finds the list of categories relevant for agile maturing. He said it is easy to recognize key challenges and main topics. He also commented that we did not find stages of evolution perhaps because we do not have enough data. He mentioned organizational ambidexterity as a relevant topic, which he and his colleagues are interested in relating to scaling agile methods. He added that in his research he recognizes some of the descriptions we made of the framework, such as losing outcomes when a team member leaves the team, for example.
Locality	He sees agile maturity in his context similarly to some of my conclusions; for example, mature teams customize and leave some agile practices. However, he did not provide clear evidence of their applicability to the Norwegian context.
Respondent 12 – Norway – Researcher – 20 years of experience in agile methods	
Utility	His opinion is that the results we presented are broad and analysis of each of the categories could be deepened. For example, just for team and organization, there would be research work for a whole lifetime. There are a number of future research studies that could be drawn up from these results.
Quality	His first comment was that we should be careful about the terms “maturity” and “process improvement” because they have been contaminated by CMMI concepts. It might create some communication problems. He also commented that the word “agile” seems to be contaminated. People think that to be agile one has to implement a list of practices. Then, he prefers to use, for example, the term “flexibility”. He also said there is nothing wrong with the framework but there is the challenge of going further with enough detail. For example, sensemaking, ambidexterity, team and organizational theories could be used as lens to explain maturing in agile. He added that, in general, the framework is good. However, he thinks that if he analyzed each of the outcomes in detail, he might disagree on some points.
Efficacy	His opinion is that we covered a lot of ground in the framework, maybe too much. Each of the categories, according to him, could be a field of study. Nevertheless, he said he sees connections among outcomes and also cannot see maturity levels, as our purpose.
Locality	He has been studying ambidexterity for a long time and he recognizes it in agile teams. However, he did not provide clear evidence of the applicability of the framework to the Norwegian context.
Respondent 13 – Italy – Researcher – 12 years of experience in agile methods	
Utility	From the practical perspective, she thinks our results help teams to understand where they stand. For her, somehow the way we presented the categories gives teams an idea of where to go. She thinks that it would be useful to identify profiles of the cases and the paths they took for improvement. She said that maybe we could suggest, for specific contexts, paths of improvement (a list of practices, for example), based even on one case. It provides a starting point for other comparative studies.
Quality	She said that, based on our empirical result, we could suggest potential levels for improvement, so that teams know where to go next. Academically, she would like to know the logic behind the categories. “Aren’t there other important dimensions that are not covered in your results?”, she asked. It is fine that dimensions are emergent from data, but she would like to see a level above and what these dimensions suggest selecting, for example, a higher level of abstraction. Maybe in a higher level of abstraction influences among categories could emerge. She also mentioned that the “organization” category seems to be at another level of analysis. It seems to be something that influences maturity but is not directed related to it, as the other categories. It would keep the analysis level consistent.

[7] Andersin, J. 2004. TPI – a model for Test Process Improvement. Seminar on Quality Models for Software Engineering, Helsinki. Available at http://goo.gl/9JSFUQ

Efficacy	She agrees that maturity in agile is a temporary picture, in the sense that changes in environment make maturity change. She said it is a "realistic picture of reality".
Locality	She said the results look familiar to her, considering the teams she studies in Europe. She said we are talking about culture, national culture. However, many times, company culture, team culture and professional culture – especially developers' culture – influence more than national culture. Therefore, she said that, based on her experience, it does not seem that a Brazilian agile developer is so different from an Italian agile developer.

The discussions on the results of this study provided us with a grounded perception of our contributions and limitations. To sum up, the answers we described in Table 41 and Table 42 showed us that our research work:

- Regarding *utility* in practical perspective: it is useful for teams to situate themselves through guided discussions and for consultants to understand the teams they are working with. Certification seems to be an interesting topic in Brazil, but it does not seem possible, given the dynamic of agile maturity;
- Regarding *utility* in research perspective: it is useful to systematize observation of agile teams. Future work should consider identifying paths of accomplishment of outcomes in different organizational contexts;
- Regarding *quality* in practical perspective: it should consider refining naming and clarifying the meaning of categories and outcomes. We have actually reviewed categories' names in the framework, and results presented in this section already consider reviewed naming;
- Regarding *quality* in research perspective: it should suggest future studies with 1) more empirical research to confirm the outcomes; 2) more empirical work to map influence among practices and outcomes; and 3) in-depth description of the categories with theoretical foundation from other fields of study;
- Regarding *efficacy*: the concepts and dynamics we proposed seem to suit reality;
- Regarding *locality*: we do not have enough evidence to confirm it is applicable in other countries. However, our results are clearly suitable to Brazilian and similar contexts.

Next section presents our discussions on the findings.

CHAPTER 6. DISCUSSIONS

We had two specific objectives in this thesis. The first was to *define* maturity in agile software development; and the second was to investigate the *mechanisms* that teams apply to mature in agile software development. This section discusses the results, as well as presents the concerns about the validity of the research. Figure 28 summarizes the contents of this chapter and contextualizes it with the previous and the next ones.

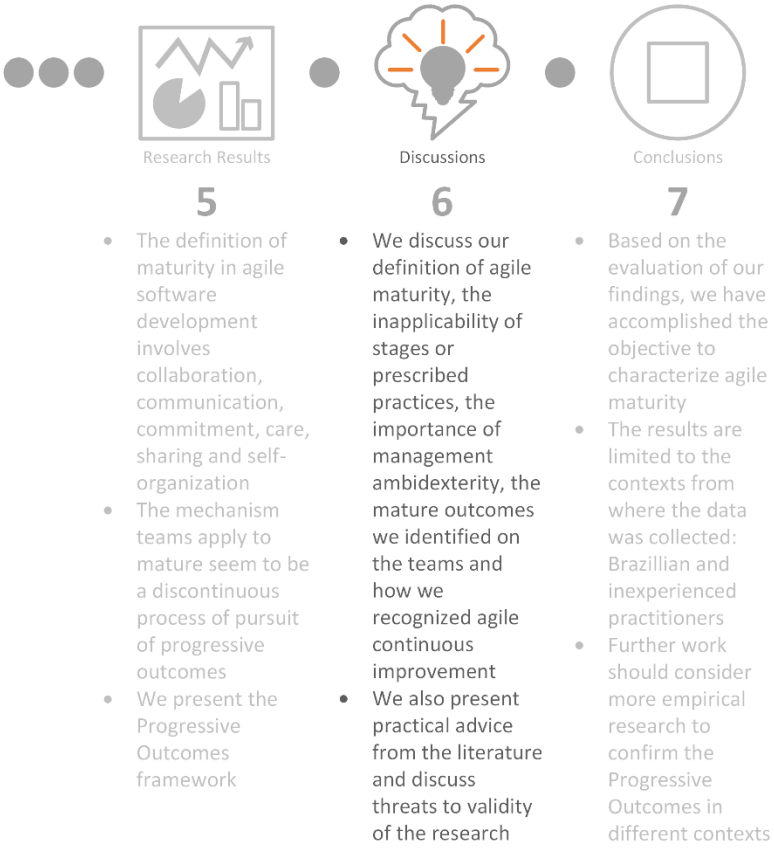


Figure 28 - Contextualization of Chapter 6

6.1 Definition of agile maturity

Our first research question aimed to investigate what maturity is in agile software development, and we have proposed such a definition based on practitioners' perceptions. As the Agile Manifesto (BECK et al., 2001) is an established reference for

agile values and practices, a comparison of the principles presented in the agile manifesto and the elements of our definition of maturity is shown in Table 43.

Some of the principles of the Agile Manifesto were not addressed by our definition of maturity, such as delivering working software frequently, integrating business people into the project, having working software as a measure of progress and simplicity at work. On the other hand, our definition adds two elements to the issues previously addressed in the Agile Manifesto: standardization of agile practices and management of source code and tests by using tools, methods and metrics. It seems that practitioners perceive the importance of having an alignment of agile practices when scaling adoption and having tools, methods and metrics to manage development to be not just agile, but also mature in agile software development.

Table 43 - Comparison of the definition of maturity with the Agile Manifesto principles

Principles of the Agile Manifesto (BECK et al., 2001)	Elements of the definition for agile software development maturity
Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.	The team generates perceived outcomes for customer and cares about customer and software quality.
Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.	The team allows requirements to change.
Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.	-
Business people and developers must work together daily throughout the project.	-
Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.	The team collaborates on projects by being committed and supported by an infrastructure for agility.
The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.	The team collaborates on projects by communicating and sharing knowledge
Working software is the primary measure of progress.	-
Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.	The team self-organizes at a sustainable pace.
Continuous attention to technical excellence and good design enhance agility.	The team cares about software quality.
Simplicity - the art of maximizing the amount of work not done - is essential.	-
The best architectures, requirements, and designs emerge from self-organizing teams.	The team self-organizes at a sustainable pace.
At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.	The team continuously improves agile practices

Although we have proposed our definition of maturity before we discovered the outcomes in the case studies, we are able to relate the definition elements to the categories of outcomes, as described in Figure 29. For example, in the second line of Figure 29: to have a team that collaborates on projects by communicating and being committed, that shares knowledge and self organizes at a sustainable pace, our investigation has shown that teams evolve from being responsive to a sparkling outcome. This evolution experiences intermediate outcomes, such as being confident

and assertive at work. Similarly, for the other elements of the definition, we show how the progressive outcomes provide support in Figure 29.

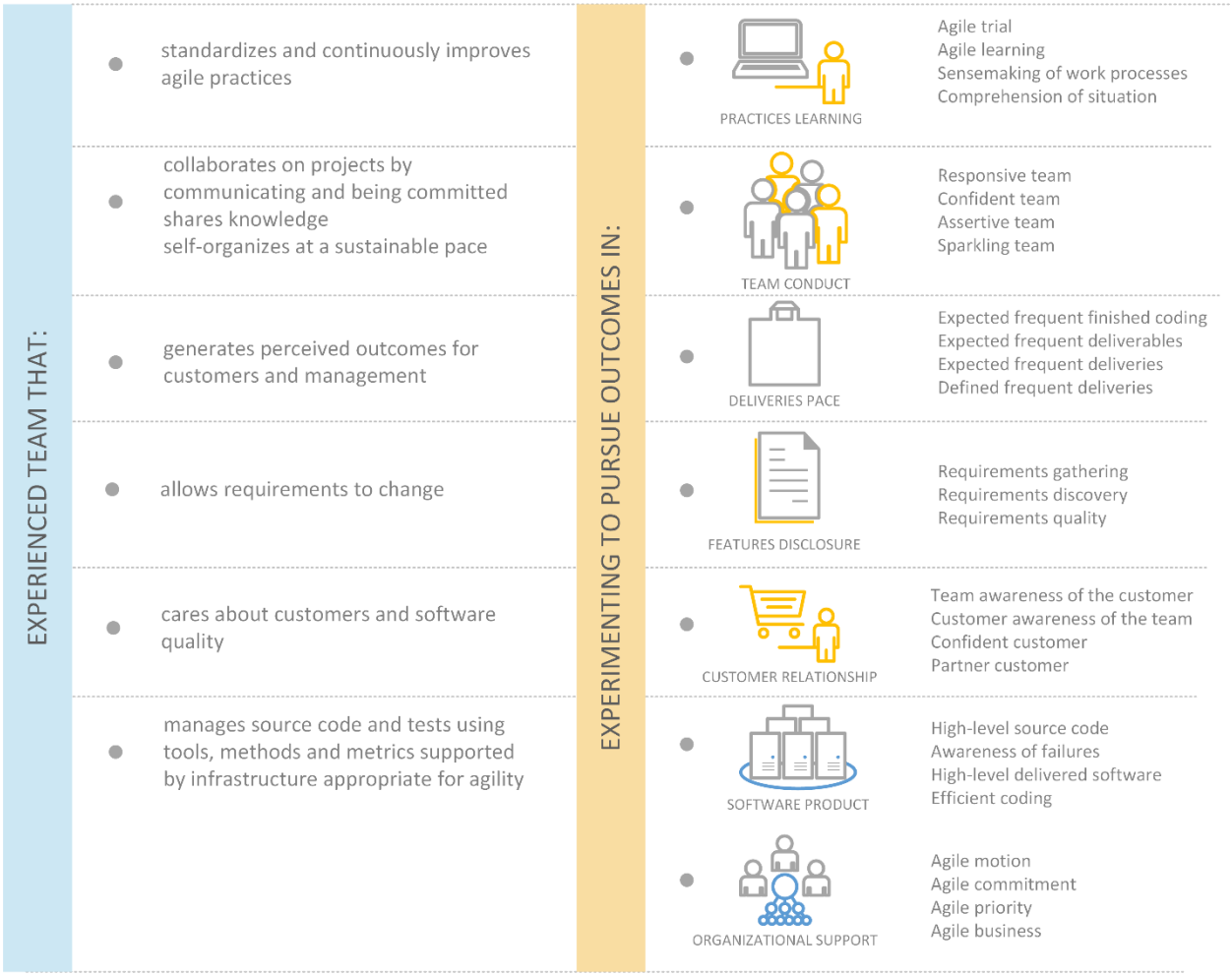


Figure 29 - Relating the definition for maturity with the Progressive Outcomes framework

6.2 Neither stages, nor prescribed practices

Our definition, as well as the framework we proposed, places an emphasis on the importance of people. This central role people play in software development is reinforced by studies which show that increasing processes definition is hard to sustain and, as time goes by, people end up working with a minimum process (COLEMAN; O'CONNOR, 2008), as software development staff focus on having the job done (ADOLPH; KRUTCHEN; HALL et al., 2012). There is plenty of evidence that, with agile methods, teams tailor their practices according to their contexts (CESARE et al., 2010; ARMBRUST; ROMBACH, 2011; SHEFFIELD; LEMÉTAYER, 2012; KIRK; TEMPERO, 2012; BUSTARD; WILKIE; GREER, 2013) and processes definition, therefore, is secondary.

For this reason, our results run counter initiatives that invest in maturing agile teams and organizations with the implementation of the CMMI-DEV reference model (as shown in CHAPTER 2). We consider these initiatives valuable if there is no concern about sustaining agility and agile values such as self-organization, simplicity and adaptability. On the other hand, if teams wish to maintain agility, they cannot rely on these reference models. Currently published agile maturity models presented in CHAPTER 2 already address this issue by proposing other means to mature in agility.

What our study adds to the current body-of-knowledge in agile maturity is that context dependence hinders the possibility of prescribing practices. Our theoretical foundation has shown that agile software development teams are complex adaptive systems. In this kind of systems, potential, creative, novel solutions emerge when the system is left to self-organize. That is, detailed codified routines hinder the emergence of optimal results. Our investigation of agile teams confirmed this lack of pattern in the adoption of the practices.

There is evidence in previous studies in the agile software development maturity field regarding the inapplicability of prescribed practices. In the model proposed by Sidky et al. (2007), for example, a set of practices was defined for each maturity level but, on the validation of the framework, practitioners have pointed out that there is a need for tailoring and considering the experiences of organizations. Kettunen (2012) had a similar conclusion, and so did the survey performed by Schweigert et al. (2012). Our survey has also indicated that agile practitioners believe that following a maturity model is valuable only if space is left for tailoring (FONTANA; REINEHR; MALUCELLI, 2014).

Accordingly, we could not identify a standardized sequence of adoption of the practices, i.e., a team implemented a practice such as “test driven development” in the beginning of agile adoption, and another team did that at a later moment. Figure 29 does not show stages or processes to be implemented. The evolvement of the practices in an agile team is a non-linear and dynamic process, in which we could identify a pattern in the outcomes pursued by the team, but not in the exact practices they endeavor to accomplish each outcome.

Indeed, complex adaptive systems work within boundaries, which constitute their attractors, but the exact form the system presents inside these boundaries cannot be predicted (STACEY, 1996). The outcomes in our framework represent the boundaries, and the practices to accomplish them should not be prescribed. It complies with the

equifinality for the development of dynamic capabilities, that is, although there are commonalities across firms, there are multiple paths to the same dynamic capability (EISENHARDT; MARTIN, 2000).

The Progressive Outcomes framework we present here (see right-hand side in Figure 29) proposes, therefore, a “semi-structure” for the maturing process, contrasting with the current agile maturity models (NAWROCKI; WALTER; WOJCIECHOWSKI, 2001; LUI; CHAN, 2005; SIDKY; ARTHUR; BOHNER, 2007; QUMER; HENDERSON-SELLERS, 2008; PATEL; RAMACHANDRAN, 2009; BENEFIELD, 2010; YIN; FIGUEIREDO; SILVA, 2011; ÖZCAN-TOP; DEMIRÖRS, 2014): there are neither stages, nor prescribed practices. It considers that software process improvement is indeed a process of emergent change, enabled and constrained by the context (ALLISON; MERALI, 2007). The emergence of the practices “is not simply a random process, but something that occurs to achieve an intended vision where the detail of that designed future is not fully understood at the time of the action” (*Ibid*., p. 678). This intended vision is what we named as pursued outcomes.

6.3 Management ambidexterity

When evolving to accomplish the outcomes, using whichever practices they feel like, teams also develop ambidextrous abilities. Ambidexterity – capacity to be simultaneously aligned and flexible (GIBSON; BIRKINSHAW, 2004) – reflects maturity in agile software development. Even before Agile Manifesto, Dybå (2000) already identified that some of the key factors for successful software process improvement were “exploitation of existing knowledge, and exploration of new knowledge” (DYBÅ, 2000, p. 259). In a study in small software development companies he warned “we should pay more attention to the inherent tensions between discipline and creativity, diversity and consensus, and knowing and doing, to mention just a few” (DYBÅ, 2000b, p. 87). At that time, he identified this ability in combining a strong skill base with experimentations as “improvisation”.

Studies in the agile software development field have responded to this claim by showing benefits in combining these dual forces of exploration and exploitation (MARCH, 1991). For example, the importance of preserving the disciplined practices when adopting agile methods (BOEHM; TURNER, 2004), as well as the established trend to combine plan-driven and agile characteristics in a single project (BASKERVILLE; PRIES-HEJE; MADSEN, 2011). The synchronization of exploration

and exploitation has already been observed in agile teams dynamics (VIDGEN; WANG, 2009), and ambidexterity, likewise (RAMESH; MOHAN; CAO, 2012; FONTANA et al., 2015b). Even on the behavior of agile practitioners, a duality has been realized: serious professionals with a free-spirit and joking behavior (HAZZAN; LERON, 2010).

Our findings provided evidence that mature teams are also ambidextrous. In accordance with the fact that there is not such a unique recipe for ambidexterity (GIBSON; BIRKINSHAW, 2004), our Progressive Outcomes framework considers this need to allow for ambidexterity development by defining the outcomes the team may pursue – the alignment – simultaneously making room for the team to implement the practices according to their contexts – the adaptability. Experimentation is essential and failure should be seen as “a natural part of process improvement" (ABRAHAMSOON; BABAR; KRUTCHEN, 2010. p. 22).

Our complementary study on ambidexterity (FONTANA et al., 2015b) has provided insights on how to accomplish it and evidenced the importance of management action to guide the team throughout the maturing process. We identified two main management strategies 1) exploiting enough to add value to the customer and the team; and 2) processes automation and results visibility. “Exploiting enough”, in the first strategy, means defining a minimum alignment on the team and leaving space for action. For example, roles are established, but the atmosphere is friendly; customer has expected and defined outcomes, but also has a relationship with the team based on trust. For the second initiative, we observed that automating processes – exploitation – generates data. These data, visible to everyone in team, foster informal and close communication – exploration. One of the worries companies have when adopting agile methods is the lack of managerial control (MELO et al., 2013b). We believe that what they fear to lose is their exploitative control. Our studies provide evidence that mature agile management is exploitative and also exploratory, keeping alignment but also fostering the adaptability that agile teams need.

6.4 Outcomes and maturity

Based on our definition of maturity in agile software development (FONTANA et al., 2014b) and on the observations made in the case studies (FONTANA et al., 2015), we propose that the outcomes accomplished by mature agile teams are the ones illustrated by Figure 30.

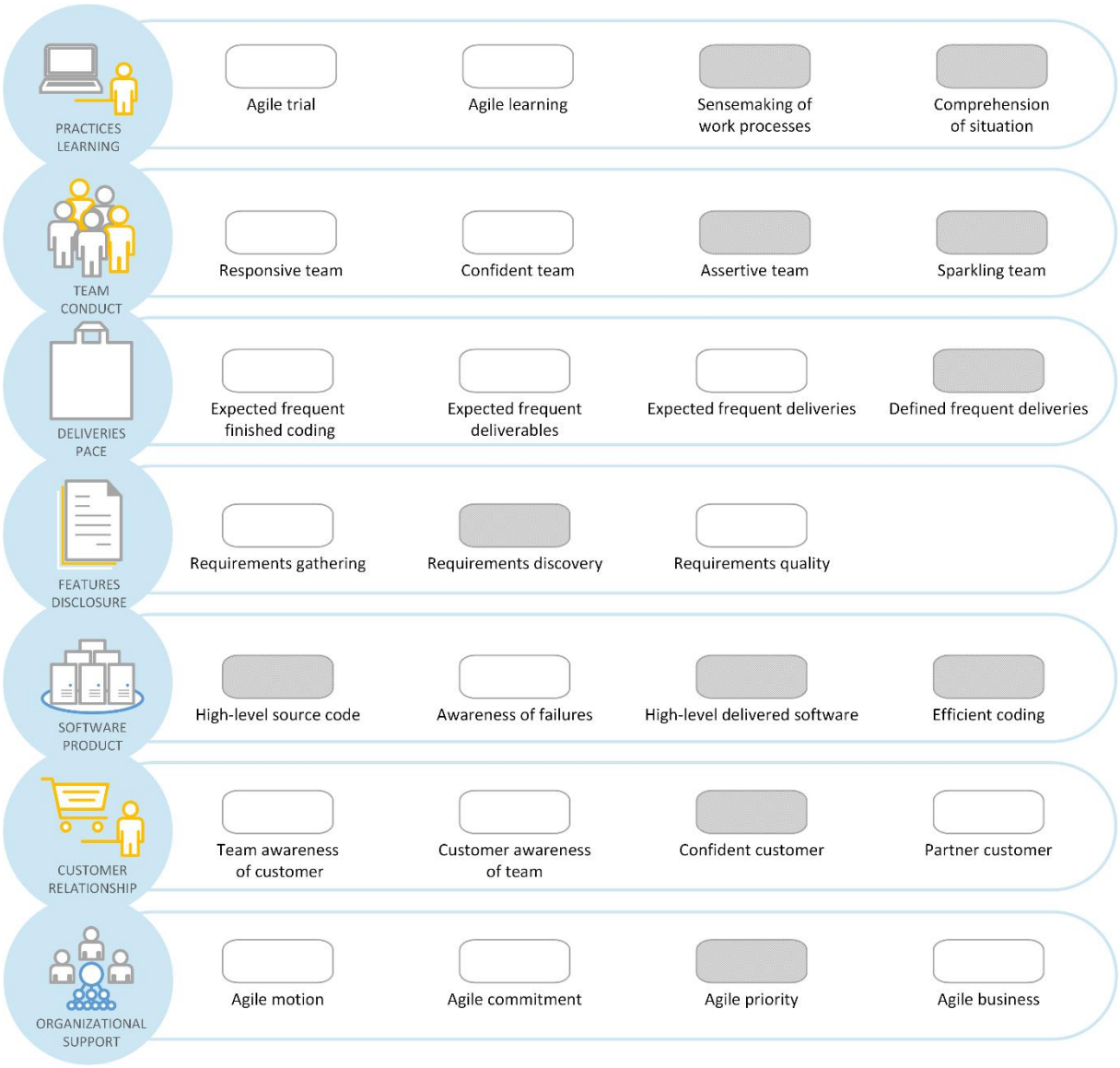


Figure 30 - Outcomes on a mature team

- As for practices learning, the team has the ability to make sense of work processes, focusing continuous action in tailoring and improvement. This is performed through a comprehension of the situation based on observations and metrics, as shown in Walter et al. (2015). Teams lack the need to follow predefined processes, but management plays an essential role here in enabling the collective perception of purpose and allowing collective decision-making. We have described some of these strategies when studying ambidextrous management in Fontana et al. (2015b).

- In team conduct there is assertiveness. The improvements made in the process are mostly proposed by the team. This team also sparks in the sense that motivation is sustained and perceived by a continuous focus on learning and technical excellence. Reflective practice is essential in this evolvement. According to Dybå, Maiden and Glass (2014), software engineers must engage in reflections about the effectiveness of the process, tools and issues involved in software engineering experience. This seems to be essential in agile maturity, since stable processes do not mean stable motivation (WALTER et al., 2015).
- Regarding deliveries pace, the team is able to deliver frequently, on time. The frequency of delivery does not matter. We had in our study teams that delivered on an hourly basis and teams that delivered on a monthly basis. They were both mature in the sense that they planned frequent deliveries and could actually do them. It should be noted that some teams adopt a lean-way to deliver, leaving sprints and adopting a continuous flow of tasks (WANG; CONBOY; CAWLEY, 2012). Even with a defined delivery date to the customer, tasks assume a continuous flow in the system.
- Features disclosure is performed with discovery of requirements during the project. Teams that accomplished this outcome have the ability to define stories (or pieces of requirements) to be delivered without change. However, if new issues emerge during development – and are recognized as priority – they are accommodated in the current sprint. It seems to be associated with the outcome of Confident Customer. The dynamics of allowing requirements to emerge and the changes new requirements trigger in prioritization of deliveries are supported and endorsed by a confident customer.
- A high-level software product is the value added to the team, an essential stakeholder in agile projects (ABRAHAMSSON; BABAR; KRUTCHEN, 2010). Mature agile teams emphasize technical excellence. They have – and “sparkle” for – high-level source code, high-level delivered software and efficient coding. They have informal and disciplined processes to assure software quality. The infrastructure for development is essential

to accomplish this outcome, and it is closely associated with the outcomes in the Organizational support category.

- In the relationship with the team, the customer must be confident. The customer provides support for frequent deliveries and requirements discovery. It became clear in the research that a team achieves a confident customer after a process of getting to know each other. The outcome in this category only applies to cases where the customer remains the same in different projects. Otherwise, there is always a new customer, and always a new acknowledgement process to start.
- Organizational support is essential for the agile evolvement process. Particularly in traditional companies, the evolution of the engagement of the organization is essential to provide the infrastructure the team needs to evolve their outcomes; thus, agile must be a priority. Although we have not explicitly investigated the organizational aspects on the teams in small companies, some of them are clearly in the "agile business" situation, because these companies were created to be agile. Here, there is a subtle intersection with the known extensions of agile values to management and Information Technology in general, as expressed in Dening (2010) – the Radical Management, and Bell and Orzen (2011) – the Lean IT. In big companies, when extending agile to business is not possible, we believe maturity is also possible when the outcome Agile priority was accomplished and strategies for coexistence with traditional management have been adopted (as described in WAARDENBURG; VLIET, 2013).

None of these outcomes can be pursued and accomplished without effective management (MELO et al., 2013b). Such management has the ability to keep a team aligned and at the same time flexible enough to reach outcomes. This ability is present in day-to-day decision-making and it seems to be strongly founded on concrete visibility of the current situation. This visibility allows the emergence of emotions that make people make sense of this situation. The use of simple metrics is essential to quickly understand the real situation and avoid making decisions based on assumptions. Simple metrics provide fast data to take simple actions that create visibility. The cycle then restarts. Figure 31 shows this management strategy, which has shown to be

effective in ambidextrously managed agile teams. The case described in Walter et al. (2015) shows examples of management actions in each of the steps in this cycle.

This cycle empirically applies Daniel Kolb's cycle of experiential learning, presented by Dybå, Maiden and Glass (2014) as a tool for reflective practice. His cycle comprises four stages: the first – concrete experience – is the actual experience to deal with situations; next comes the reflection and observation of the experiences from different point of views – the reflective observation; the third stage is abstract conceptualization, in which individuals make sense of the experiences, explaining what has been observed; and the last stage – active experimentation – is when individuals test their ideas trying something new.

What happens after all outcomes have been accomplished? If the team is in a stable context (no changes in team, customer, organization), focus is on specific improvements, on technical excellence and on sustaining outcomes through continuous improvement.

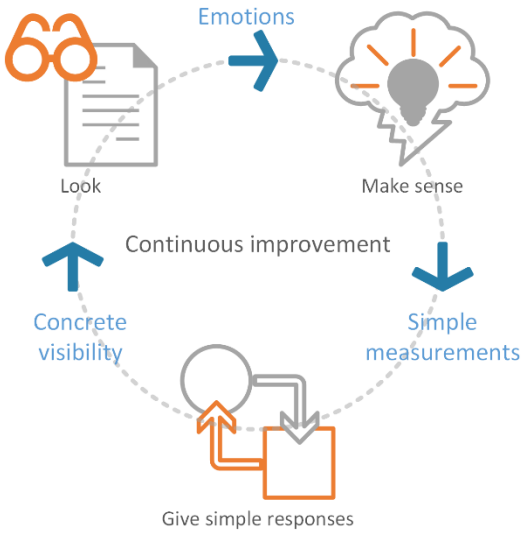


Figure 31 - Management strategies for continuous improvement

6.5 Continuous improvement

Continuous improvement is also the aim of high-maturity teams which have adhered to CMMI-DEV. The difference from the context of agile teams is that continuous improvement is not founded on extensive processes definition and measurement. They do not necessarily have to define work, then control it, then improve it, in this order. As we described earlier in the management strategy, the continuous improvement cycle is quick (look, make sense and give simple responses) and performed since the very beginning of the work on the team (CLARKE;

O'CONNOR, 2011). Our study confirmed early conclusions from Dybå (2000) in the sense that, to accomplish improvements in software processes, commitment to learning is more important than to "best practice" models. We understand that this visibility-based continuous improvement leads to an ambidextrous behavior, which characterizes maturity in agile software development. Figure 32 represents how we place continuous improvement in the maturing process with agile methods, in comparison with continuous improvement in a CMMI-DEV maturing process.

Mature teams in agile software development are therefore different from mature teams which follow CMMI-DEV guidelines. It reflects the fact that our research is founded on a theoretical basis different from that of CMMI-DEV. CMMI-DEV guidelines for maturity are founded on lessons learned from manufacturing. This is the context where stability and repeatable processes are desirable (DYBÅ, 2000b). The example given by Poppendieck and Poppendieck (2003) is valuable here. They differ the process of creating a new recipe from making the dish. Creating a recipe is a creative process that does not benefit from repeatability. It benefits from allowing experimentation and change. However, when a recipe is created, making the dish is actually a process where repeatability and reduction of variation are necessary to reach efficiency and quality. As we consider to be working with complex adaptive systems, our findings best fit maturity for creating new recipes. We agree with the learning perspective from Dybå, Maiden and Glass (2014): "the conventional way of thinking about software process improvement puts [...] in a sequential order - first understanding, then action. [...] From a learning perspective, however, actions and understandings often need to be reversed" (DYBÅ; MAIDEN; GLASS, 2014, p. 34).

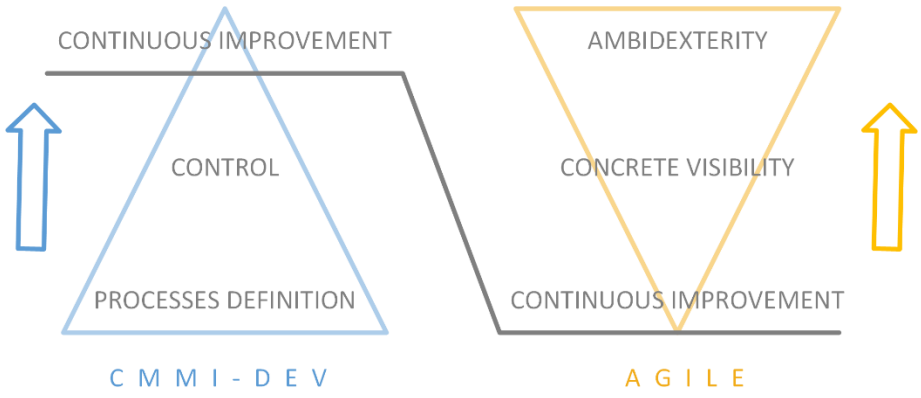


Figure 32 - Comparison of continuous improvement in CMMI-DEV and in agile methods

6.6 A temporary picture

The outcomes a team accomplished during the maturing process may change as long as the environment changes: outcomes on a team recede if members are replaced, outcomes in a product recede if technology changes, outcomes in a customer recede if customer changes, outcomes in an organization recede with changes in business strategy.

This situation confirms our theoretical foundation of complex adaptive systems: there is no stability – but edge-of-chaos, instead. It highlights two important issues: 1) the management strategies that will sustain improvement even with instability and 2) the impossibility to label maturity with a tag. How can we label a team with alleged agile maturity, if, in the case of team member change, the team will lose maturity?

Nevertheless, the diagnosis of agile maturity seems useful for its primary purpose (KOHLEGGER; MAIER; THALMANN, 2009): situate the team and give clues for improvement. Our framework, complemented by the Agile Compass, does not provide a maturity tag; thus, it does not instigate ranking. The contribution is allowing teams to situate themselves and glimpse where new improvements should take place, with a clear view that there is a learning process and management strategies to be followed. It addresses a need Dybå, Maiden and Glass (2014) identified when they say that developers and software projects lack information that lead to a reflective practice: “Not only to software developers lack the tools to capture, analyze and present information upon which to reflect, most software projects don't actively support reflection, or budget or schedule for it”. (DYBÅ; MAIDEN; GLASS, 2014, p. 32).

6.7 Practical advice from literature

Current research in agile methods provides practical evidence that aids teams in finding guidance to pursue outcomes in agile practices learning, team conduct, deliveries pace, features disclosure, software products, customer relationship and organizational support, as shown in Table 44.

Table 44 - Insights from recent studies on the outcomes agile teams pursue

Category of Outcome	Insights from recent studies
PRACTICES LEARNING	• Power and Conboy (2014) studied the impediments to flow in agile methods. These impediments are "anything that obstructs the smooth flow of work through the system and/or interferes with the system achieving its goals" (p. 206). Four out of the nine impediments they identified are related to practices: delays (people waiting for things to happen); failure demand; work in progress; and extra processes. • Gandomani et al. (2014) found that training is not only a significant driver in the agile transformation process but it is also, in many cases, inadequate and dysfunctional. The main reasons they identified for inadequate

	and dysfunctional training are: partial training, inappropriate contents and non-practical training, time-boxed training, lack of time commitment (to the training), and human aspects (resistance to change). • One of the worries companies have when adopting agile methods is the lack of management control (MELO et al., 2013). Our study has shown that it is addressed in practice by exploitative behavior. • The more experienced the company, the more agile practices it adopts, and they sometimes give up practices because of context variables of the project, not because practices are useless (MELO et al., 2013). • The iteration retrospective is an important moment in the projects, as short-term improvements in processes are usually discussed in this meeting (DRURY; CONBOY; POWER, 2012).
TEAM CONDUCT	• According to Melo et al. (2013) 91% of the companies in Brazil that adopted agile methods did so to increase team productivity. On the other hand, one of the main reasons for agile adoption failure is lack of experience in agile methods (MELO et al., 2013), which is evidence of the importance of the team in agile methods evolvement. • Four out of the nine impediments to flow that Power and Conboy (2014) identified are related to people issues, such as, handovers, context switching, unnecessary motion, unmet human potential. • Kettunen (2014) says that agile enterprises can be improved by fostering teams gauging their performance accordingly towards the desired states: "since software work processes are actually performed by teams (and their individuals), the transformations are ultimately realized by addressing them." (p. 291). • Adolph et al. (2012) identified that agile teams focus on having the job done, and it includes satisfying the software team members' needs to reach the end and achieve a feeling of accomplishment. The team also needs to see technical debt minimized. • Drury-Grogan (2014) identified that satisfying teams is one of the objectives that teams pursue during iterations. • Developers being pulled to deal with customer support issues during iteration may negatively affect decision-making in agile projects (DRURY; CONBOY; POWER, 2012). • Choices of the team design are an important factor that impacts team productivity. Melo et.al. (2013b) identified that full time team members contribute to the team focus; in mixed team members, novice members contribute flexibility and experienced ones contribute knowledge; small teams lead to better communication, conflict management, commitment and sense of responsibility; and team collocation helps in negotiation and planning of requirements. • The study by Moe et al. (2010) provides valuable insights on the importance of communication, feedback and trust. • The experience of the team contributes to the emergence of practices in shadow networks (STACEY, 1996), as more experienced individuals usually create an increased number of informal social networks and focus their energy on productive discussions (ADOLPH; KRUTCHEN; HALL et al., 2012). • Moe, Dingsøyr and Dybå (2009) suggest, to foster self-managing teams, building redundancy and team-level by: organizing cross-training, collocating the team at the same room, appreciating generalists practitioners, building trust and commitment, assigning people to one project at a time.
DELIVERIES PACE	• Melo et al. (2013) identified that 73% of Brazilian companies that adopted agile methods sought to accelerate time to market, that is, speed-up deliveries. • Olsson and Bosh (2014) say that the final stage of agile adoption is when deliveries are “experiment systems” because feedback from the customer is collected real-time, instead of “being frozen early as part of requirements prioritization” (p. 329). They also see continuous deployment as the “heaven” of adoption of agile methods practices. • Drury-Grogan (2014) identified that meeting schedule is one of the objectives that teams pursue during iterations. • Bellomo (2013) emphasizes the importance of architecture definitions and prototyping for rapid delivery. Successful teams weave architecture and requirements during prototyping and develop flexible architectures. • Tonelli et al. (2013) identified that some factors affect accelerating deliveries, such as planning sprints with bug correction, and better product development guided by test/test-driven development with instant feedback, among other factors.
FEATURES DISCLOSURE	• Melo et al (2013) identified that managing changing priorities is the reason for adopting agile methods in 86% of agile companies in Brazil. • One impediment to flow identified by Power and Conboy (2014) is related to requirements: including extra features in requirements. • Drury-Grogan (2014) identified that meeting functionality is one of the objectives that teams pursue during iterations. She also identified that accepting iterations amendments after sign-off (adding additional work to the sprint) is one issue that affects project management success. • The main physical artifacts in agile software development are story cards and the wall and, when teams substitute these artifacts by other means, the notational and social perspectives may be harmed (SHARP; ROBINSON; PETRE, 2009). • The need to balance between the need for progress (user requirements) and for quality (team requirements) may lead to problems in software quality and code disorganization (MOE; AURUM; DYBÅ, 2012). • Wnuk et al. (2013) verified that requirements identified by domain experts are less likely to become obsolete during the project and, thus, show the importance of team expertise about the customer business.

SOFTWARE PRODUCT	• Melo et al. (2013) identified that one of the main reasons why companies adopted agile methods was to enhance software quality. • Architecture definition in agile software is an essential activity because it seeks to provide product quality and deliver value to a key class of stakeholders. i.e., developers. The architectural issues must be defined as soon as possible in the project life cycle, and documentation must be used as needed: in most cases, an architectural prototype will suffice, but in others, more explicit documentation is needed (ABRAHAMSSON; BABAR; KRUTCHEN, 2010). • Drury-Grogan (2014) identified that delivering quality products is one of the objectives teams pursue during iterations. • In the dynamic environment of agile approaches, the continuous modification of software can cause risks for software developers, as code may become fault-prone or difficult to maintain. Thus, Antinyan et al. (2013) suggest that risk in modifying code should be assessed through metrics, such as number of revisions in the code and cyclomatic complexity. • Technical debt is one of the problems created with frequent and timeless coding changes, and Krishna and Basu (2012) suggested a number of practices to reduce technical debt, e. g. developers should smell their own codes and take help from others. • Collins and Lucena (2012) investigated the adoption of automated tests in agile teams and identified that experimenting is a key ability: teams should experiment new practices and technologies and abandon them when they do not feel they are useful. Some of their findings was that test automation must be simple and address tests that add value to the product.
CUSTOMER RELATIONSHIP	• According to Melo et al. (2013), 72% of agile Brazilian companies adopted agile methods to improve alignment between IT and business. • Adolph et al. (2012) identified that agile teams focus on having the job done, and it includes creating software that appeases the customer. • Customer focus on agile projects may be characterized by customer knowledge, customer involvement, customer identity, customer characteristics, customer location and team experience with the customer (LOHAN; LANG; CONBOY, 2011). • Customer involvement is critical to agile projects. Hoda et al. (2011) identified that inadequate customer involvement leads to adverse consequences for self-organizing agile teams. These consequences were pressure to over-commit, problems with gathering and clarifying requirements, problems with prioritizing requirements, problems with securing feedback, loss of productivity, and in extreme cases, business loss.
ORGANIZATIONAL SUPPORT	• Dybå (2000) emphasizes organizational role in software process improvement initiatives in general affirming that “process improvement cannot be managed, but only enabled through the space in which the software organization creates the possibilities for sensemaking, knowledge creation, and purposeful action” (DYBÅ, 2000, p. v). • Manen and Vliet (2014) identified that the “agile mindset” is crucial for the expansion of agile methods across the organization. They identified the issues that are related to enabling agile mindset: collaboration, trust and continuous improvement. For each of these factors, they show the factors that influence each of these issues, either positively or negatively. • Communities of practice play a significant role in successful agile transformation. Communities of practice is a group of practitioners who wish to deepen their knowledge in a common topic. Paasivaara and Lassenius (2014) presented a case in a large organization where communities of practice were used and recognized as essential for agile transformation and, also, after the method was established, for continuous improvement. • Waandenburg and Vliet (2013) studied the challenges of co-existence of agile methods with plan-driven approaches in an organization. They identified that these challenges are classified in two categories: increased IT landscape complexity and lack of business involvement. They presented the conditions, causes, consequences and contingences for the challenges. • Paasivaara et al. (2014) describe how “value workshops” helped Ericsson to align different sites and teams in agile transformation. The company decided to create workshops, which were held by agile coaches and a few managers, to create a vision of where the organization was heading for. Five core values were transmitted during the workshops: one organization, step-by-step, customer collaboration, passion to win and fun. The workshops also had the objective to create collaboration among sites and a “we” spirit. • Santos et al. (2014) presented a model for effective knowledge sharing among teams when agile has scaled to the whole organization. Their model emphasizes that inter-team knowledge sharing is only effective when the process is completely implemented, rather than only specific practices. Those practices are mainly related to socializing knowledge and their effectiveness is influenced by organizational context and stimuli. • Moe, Dingsøyr and Dybå (2009) identified that the barriers to a self-managing are not just in the team itself, but also in the organizational context. Team-level barriers they identified were individual commitment, individual leadership and failure to learn. On the organizational level, the barriers to self-managing software teams were shared resources, organizational control and specialist culture.

6.8 Threats to validity of research

The first threat to the validity of this study is the sample size in the survey in which we investigated the definition for maturity in agile software development. The sample size is a problem of statistical significance. The data were collected from a sample of fifty-one individuals in a population from which we cannot estimate the size and, thus, our results may be valid only for our sample, and not for the whole population (LEE; MOHAREJI, 2012).

There is, however, a need to differentiate statistical significance from practical relevance (KITCHENHAM et al., 2002). Practical relevance concerns generating findings that matter to practitioners, which is especially important if one considers that software engineering is an applied discipline and that the results of any research must be of interest to the software industry (SJØBERG et al., 2008).

We have not confirmed the statistical significance of the survey, but we consider that our findings have “conceptual relevance”, as it is scientific knowledge that affects how we perceive a decision situation or modifies our understanding of a decision situation (NICOLAI; SEIDL, 2010). One way to identify practical relevance in a study is to conduct further empirical studies on the same subject, which was performed in this dissertation.

Regarding the case studies, construct validity and external validity are the main concerns. Construct validity consists in verifying the operational measurements for the concepts that are being studied (YIN, 2005). One threat to the construct validity in our study was the use of narratives to identify agile practices evolvement. We recognize that the narrative approach for research is compatible with the need to appreciate the complexity of organizations (TSOUKAS; HATCH, 2005). However, we had to rely on interviewees’ memories to trace the agile practices adoption and there might be gaps in their stories. This is the reason why we conducted more than one interview in each team, so that one story could complement the others.

With respect to external validity, which is the ability to generalize results, we sought for analytical generalization, instead of statistical generalization (YIN, 2005; SJØBERG et al., 2008). We used the replication logic in our multiple-cases study (YIN, 2005) and grounded our data analysis on a conceptual framework to reinforce evidence for external validity. Moreover, evaluations were conducted with members of the agile community in Brazil and in Europe to evaluate the validity of the findings (see

Figure 9). Nevertheless, we consider our findings to be subject to validation or refutation by further studies.

CHAPTER 7. CONCLUSIONS

This manuscript described the results we obtained in our study to characterize maturity in agile software development. Based on the theoretical foundation of the complex adaptive systems theory, our research was organized into three main stages: Stage 1 – in which we defined maturity in agile software development; Stage 2 – in which we could understand the mechanisms to mature with agile methods; and Stage 3 – in which the findings were evaluated. Figure 33 summarizes how the rationale has been developed throughout the thesis.

Figure 33 - Thesis summary

It is been fifteen years since the publication of the Agile Manifesto (BECK et al., 2001). We are, therefore, currently experiencing the maturing of the agile software development movement in a way that is has been considered to be the main stream in some parts of the world (BUSTARD; WILKIE; GREER, 2013). In Brazil, the adoption of such methods is emerging (MELO et al., 2013). The practical relevance in our research is that, while the agile adoption has grown and continues to grow, teams that have implemented these methods are maturing their practices. It is the moment to

investigate how they are accomplishing their results and create guidelines to help other teams to evolve in agility.

The academic relevance of the study lies in the evidence that we have demonstrated on the variety of existing agile maturity models, and the variety of underlying concepts. “Agile maturity” is still being defined and, as such, it was considered one of the trend areas in agile research at the 15th International Conference in Agile Software Development (FALESSI et al., 2014).

Our contributions with this study are: a definition for agile software development maturity that considers the nature of the agile teams, and a framework that describes agile software development evolvement, complemented with a diagnosis tool. The Progressive Outcomes framework that we proposed considers people as the central role in the maturing process, sees ambidexterity as a key ability to maturity, and does not prescribe practices, but outcomes that teams actually pursue. Agile maturity comes from a non-linear and dynamic process of pursuit of progressive outcomes in the practices learning, on the team conduct, in the deliveries pace, in the way features are disclosed, in the quality of the software product, in customer relationship and in organizational support.

We cannot assume that these findings would speed up agile transformation or make the evolvement path easier. The feedback we have received from practitioners showed us that our main contributions are 1) showing that agile improvement is beyond learning of practices; instead, it involves a number of issues (our categories of outcomes) that need to be addressed; 2) showing that learning is the focus, as evolvement is dynamic and based on continuous improvement since the beginning; and 3) emphasizing the importance of ambidextrous management to guide an adaptive maturing process.

Our results are limited to the contexts where data were collected. Firstly, to the Brazilian agile adoption context and, secondly, to the specific teams we have investigated. The Brazilian context of agile adoption is that of inexperienced practitioners. To mitigate this limitation, we have conducted validations with researchers from other countries. Although to some of them our results seemed familiar, we cannot assure applicability abroad. The limitation to the context of the teams we analyzed is a known limitation of case studies (EISENHARDT, 1989), which deepen the analysis, but do not provide an extensive sample. We have worked to create a sample of companies with different profiles but, still, our findings are subject

to replication in other contexts. Our findings are also limited to the evolution of agile practices within teams and, therefore, they do not address scaling of agile practices across team boundaries (WAARDENBURG; VLIET, 2013).

Further work of this study might include action research to implement and test the diagnosis tool we proposed in multiple environments. Moreover, our evaluation of the results of this dissertaion emphasized the need for: 1) more empirical research to confirm our outcomes in different contexts; 2) more empirical work to identify evolvement paths of outcomes accomplishment in different contexts; 3) more empirical work to map influence among practices and outcomes; and 4) studies that deepen the description of the categories with theoretical foundation from other fields of study.

REFERENCES

ABBAS; GRAVEL; WILLS, 2010	ABBAS, N., GRAVELL, A. M., WILLS, G. B. 2010. Using Factor Analysis to Generate Clusters of Agile Practices - A guide for agile process improvement. In: Proceedings of the Agile Conference (2010), pp. 9-13. DOI: 10.1109/AGILE.2010.15
ABRAHAMSSON; BABAR; KRUTCHEN, 2010	ABRAHAMSSON, P.; BABAR, M. A.; KRUTCHEN, P. 2010. Agility and Architecture: Can They Coexist? IEEE Software. Vol. 27. No. 2. pp. 16 - 22. DOI: 10.1109/MS.2010.36
ABRAHAMSSON et al., 2003	ABRAHAMSSON, P; WARSTA, J.; SIPONEN, M.; RONJAINEN, J., 2003. New directions on agile methods: a comparative analysis. ICSE '03: Proceedings of the 25th International Conference on Software Engineering. 3-10 May. pp. 244-254. DOI 10.1109/ICSE.2003.1201204.
ABRANTES; TRAVASSOS, 2013	ABRANTES, J. F.; TRAVASSOS, G. H. 2014. Towards Pertinent Characteristics of Agility and Agile Practices for Software Processes. CLEI Eletronic Journal. Vol. 16. No. 1, Available in http://www.scielo.edu.uy/scielo.php?script= sci_arttext&pid=S0717-50002013000100006&lng=es&nrm=iso
ADOLPH; KRUTCHEN; HALL et al., 2012	ADOLPH, S.; KRUTCHEN, P.; HALL, W. 2012. Reconciling perspectives: A grounded theory of how people manage the process of software development. The Journal of Systems and Software, 85, 1269-1286. DOI: 10.1016/j.jss.2012.01.059
AGRAWAL; CHARI, 2007	AGRAWAL, M.; CHARI, K. 2007. Software Effort, Quality, and Cycle Time: A Study of CMM Level 5 Projects. IEEE Transaction on Software Engineering, vol. 33, no. 3, pp. 145-156. DOI: 10.1109/TSE.2007.29
ALLISON; MERALI, 2007	ALLISON, I.; MERALI, Y. 2007. Software process improvement as an emergent change: A structurational analysis. Information and Software Technology. Vol. 49. pp. 668-681. DOI: 10.1016/j.infsof.2007.02.003
AL-TARAWNEH; ABDULLAH; ALI, 2011	AL-TARAWNEH, M. Y., ABDULLAH, M. S., ALI, A. B. 2011. A proposed methodology for establishing software process development improvement for small software development firms. Procedia Computer Science, 3, pp. 893-897. DOI: A proposed methodology for establishing software process development improvement
ANDERSON, 2005	ANDERSON, D. J., 2005. Stretching Agile to fit CMMI Level 3 – the story of creating MSF for CMMI Process Improvement at Microsoft Corporation. Proceedings of the Agile Conference

(ADC'05). 24-29 July. pp. 193-201. DOI 10.1109/ADC.2005.42.

ANTINYAN et al., 2014 — ANTINYAN, V.; STARON, M. ; MEDING, W. ; OSTERSTROM, P. ; WIKSTROM, E. ; WRANKET, J. ; HENRIKSSON, A. ; HANSSON, J. 2014. Identifying risky areas of software code in Agile/Lean software development: An industrial experience report. Proceedings of the IEEE Conference on Software Maintenance, Reengineering and Reverse Engineering. Antwerp. 3-6 Feb. pp. 154 - 163. DOI: 10.1109/CSMR-WCRE.2014.6747165

ARMBRUST; ROMBACH, 2011 — ARMBRUST, O.; ROMBACH, D. 2011. The right process for each context: objective evidence needed. ICSSP '11: Proceedings of the 2011 International Conference on Software and Systems Process. May. pp. 237-241. DOI: 10.1145/1987875.1987920

AUGUSTINE et al., 2005 — AUGUSTINE, S., PAYNE, B., SENCINDIVER, F., WOODCOCK, S. 2005. Agile Project Management: Steering from the edges. Communications of the ACM, Vol. 48, No. 12, pp. 85-89. DOI: 10.1145/1101779.1101781

AXELROD; COHEN, 2000 — AXELROD, R., COHEN, M. D. 2000. Harnessing Complexity: Organizational Implications of a Scientific Frontier. New York: The Free Press.

BAKER, 2006 — BAKER, S. W., 2006. Formalizing Agility, Part 2: How an Agile Organization Embraced the CMMI. Proceedings of the AGILE 2006 Conference. 23-28 July. pp. 146-154. DOI 10.1109/AGILE.2006.30.

BARDIN, 2011 — BARDIN, L. 2011. Análise de Conteúdo. São Paulo: Edições 70.

BASKERVILLE; PRIES-HEJE; MADSEN, 2011 — BASKERVILLE, R.; PRIES-HEJE, J.; MADSEN, S. 2011. Post-agility: What follows a decade of agility? Information and Software Technology, Vol. 53, No. 5, pp. 543-555, DOI: 10.1016/j.infsof.2010.10.010

BECK et al., 2001 — BECK, K. et al., 2001. Agile Manifesto. Available in http://agilemanifesto.org/. Accessed in 2013, May.

BELL; ORZEN, 2011 — BELL, S.; ORZEN, M. 2011. Lean IT: Enabling and Sustaining Your Lean Transformation. Flórida: CRC Press.

BELLOMO, 2013 — BELLOMO, S. 2013. Elaboration on an integrated architecture and requirement practice: Prototyping with quality attribute focus. Proceedings of the 2nd International Workshop on the Twin Peaks of Requirements and Architecture (TwinPeaks). San Francisco, CA. 21-21 May 2013. pp. 8-13. DOI: 10.1109/TwinPeaks.2013.6614717

BENEFIELD, 2010 — BENEFIELD, R. 2010. Seven Dimensions of Agile Maturity in the Global Enterprise: A Case Study. Proceedings of the 43rd

Hawaii International Conference on System Sciences. pp. 1-7. Honolulu, HI. DOI 10.1109/HICSS.2010.337.

BLAND; ALTMAN, 1997 — BLAND, J. M.; ALTMAN, D. G. 1997. Statistics notes: Cronbach's alpha. BMJ. Vol. 314. pp. 572. DOI 10.1136/bmj.314.7080.572.

BOEHM; TURNER, 2004 — BOEHM, B., TURNER, R. 2004. Balancing Agility and Discipline: Evaluating and Integrating Agile and Plan-Driven Methods. Proceedings of the 26th International Conference on Software Engineering. 23-28 May. pp. 718-719. DOI: 10.1109/ICSE.2004.1317503

BONESSO; GERLI; SCAPOLAN, 2014 — BONESSO, S.; GERLI, F.; SCAPOLAN, A. 2014. The individual side of ambidexterity: Do individual's perceptions match actual behaviors in reconciling the exploration and exploitation trade-off? European Management Journal. Vol. 32. No. 4. pp. 392-405. DOI: 10.1016/j.emj.2013.07.003. 2014.

BOURQUE; FAIRLEY, 2014 — BOURQUE, P.; FAIRLEY, R. E. (Eds.) 2014. Guide to the Software Engineering Body of Knowledge, Version 3.0, IEEE Computer Society. Available at www.swebok.org.

BRYMAN, 2012 — BRYMAN, A. 2012. Social research methods. 4th edition. New York: Oxford University Press.

BUGLIONE, 2011 — BUGLIONE, L. 2011, June. Light Maturity Models (LMM): an Agile application. Profes '11: Proceedings of the 12th International Conference on Product Focused Software Development and Process Improvement. pp. 57-62. DOI: 10.1145/2181101.2181115

BUSTARD; WILKIE; GREER, 2013 — BUSTARD, D.; WILKIE, G.; GREER, D. 2013. The Maturation of Agile Software Development Principles and Practice: Observations on Successive Industrial Studies in 2010 and 2012. 20th Annual IEEE International Conference and Workshops on the Engineering of Computer Based Systems (EBCS). Apr 22-24. pp. 139-146. DOI 10.1109/ECBS.2013.11.

CAFFERY; PIKKARAINEN; RICHARDSON, 2008 — CAFFERY, F. M., PIKKARAINEN, M., RICHARDSON, I. 2008. AHAA – Agile, Hybrid Assessment Method for Automotive, Safety Critical SMEs. ICSE '08: Proceedings of the 30th international conference on Software engineering. pp. 551-560. DOI: 10.1145/1368088.1368164

CAMPBELL-HUNT, 2007 — CAMPBELL-HUNT, C. 2007. Complexity in practice. Human Relations, Vol. 60, No. 5, pp. 793-823. doi:10.1177/0018726707079202

CESARE et al., 2010 — CESARE, S.; LYCETT, M.; MACREDIE, R. D.; PATEL, C.; PAUL, R. 2010. Examining Perceptions of Agility in Software Development Practice. Communications of the ACM, Vol. 53, No. 6, pp. 126-130, DOI: 10.1145/1743546.1743580

CILLIERS, 2002	CILLIERS, P. 2002. Why we cannot know complex things completely. Emergence. Vol 4. No. 1-2. pp. 77-84. DOI: 10.1080/15213250.2002.9687736
CLARKE; O'CONNOR, 2011	CLARKE, P.; O'CONNOR, R. V. 2011. An Approach to Evaluating Software process Adaptation. In: R. V. O'Connor (Eds.), Proceedings of the 11th International Conference, SPICE 2011, Dublin, Ireland, May 30 – June 1, pp. 28-41, DOI: 10.1007/978-3-642-21233-8_3
CMMI Product Team, 2010	CMMI Product Team. 2010. CMMI for Development, Version 1.3 (CMU/SEI-2010-TR-033). Software Engineering Institute, Carnegie Mellon University. Available at http://www.sei.cmu.edu/library/abstracts/reports/10tr033.cfm
COCKBURN, 2007	COCKBURN, A. 2007. Agile Software Development: The Cooperative Game. Second Edition. Boston: Addison-Wesley.
COHAN; GLAZER, 2009	COHAN, S; GLAZER, H. 2009. An Agile Development Team's Quest for CMMI Maturity Level 5. Agile Conference 2009. 24-29 Aug. pp. 201-206. DOI: 10.1109/AGILE.2009.24.
COLEMAN; O'CONNOR, 2008	COLEMAN, G.; O'CONNOR, R. 2008. Investigating software process in practice: A grounded theory perspective. The Journal of Systems and Software, Vol. 81, No. 5, pp. 772-784. DOI: 10.1016/j.jss.2007.07.027
COLLINS; LUCENA, 2012	COLLINS, E. F.; LUCENA, V. F. Software Test Automation practices in agile development environment: An industry experience report. 2012. Proceedings of the 7th International Workshop on Automation of Software Test (AST). Zurich. 2-3 June. pp. 57 - 63. DOI: 10.1109/IWAST.2012.6228991.
CONBOY et al., 2011	CONBOY, K., COYLE, S., WANG, X., PIKKARAINEN, M. 2011. People over process: key challenges in agile development. IEEE Software, Vol. 28, No. 4, pp. 48-57, DOI: 10.1109/MS.2010.132
DAVIES; BOULDIN, 1979	DAVIES, D. L.; BOULDIN, D. W. 1979. A Cluster Separation Measure. IEEE T Patter Anal, vol PAMI-1, Issue. 2. pp. 224 - 227. DOI 10.1109/TPAMI.1979.4766909.
DECLERCQ; THONGPAPANL; DIMOV, 2013	DE CLERCQ, D.; THONGPAPAN, N.; DIMOV, D. 2013. Shedding new light on the relationship between contextual ambidexterity and firm performance: An investigation of internal contingences. Technovation. Vol. 33. pp. 119-132. DOI: 10.1016/j.technovation.2012.12.002.
DENNING, 2010	DENNING, S. 2010. The leader's guide to radical management: reinventing the workplace for the 21st century. San Francisco: John Wiley & Sons.
DEROSA; MCCAUGHIN, 2007	DEROSA, J. K., MCCAUGHIN, L. K. 2007. Combined Systems Engineering and Management in the Evolution of Complex Adaptive Systems. Proceedings of the 1st Annual

IEEE Systems Conference, pp. 1-8, DOI: 10.1109/SYSTEMS.2007.374653

DINGSØYR et al., 2012 — DINGSØYR, T., NERUR, S., BALIJEPALLY, V., MOE, N. B., 2012. A decade of agile methodologies: Towards explaining agile software development. Journal of Systems and Software. Vol. 85. Nol. 6. pp. 1213–1221. DOI: 10.1016/j.jss.2012.02.033.

DRURY; CONBOY; POWER, 2012 — DRURY, M.; CONBOY, K.; POWER, K. 2012. Obstacles to decision making in Agile software development teams. Journal of Systems and Software. Vol. 85. No. 6. pp. 1239-1254. DOI: 10.1016/j.jss.2012.01.058

DRURY-GROGAN, 2014 — DRURY-GROGAN, M. L. 2014. Performance on agile teams: Relating iteration objectives and critical decisions of project management success factors. Information and Software Technology. Vol. 56. No. 5. pp. 506-515. DOI: 10.1016/j.infsof.2013.11.003

DYBÅ, 2000 — DYBÅ, T. 2000. Enabling Software Process Improvement: An Investigation of the Importance of Organizational Issues. Doctoral Dissertation presented to the Department of Computer and Information Science of the Norwegian University of Science and Technology. Accessed through personal communication, in June, 2015.

DYBÅ, 2000b — DYBÅ, T. Improvisation in Small Software Organizations. 2000. IEEE Software. Vol. 15. No. 5. pp 82-87. DOI 10.1109/52.877872

DYBÅ; DINGSØYR, 2008 — DYBÅ, T.; DINGSØYR, T. 2008. Empirical studies of agile software development: A systematic review. Information and Software Technology. Vol. 50. No 9-10. pp. 833–859. DOI 10.1016/j.infsof.2008.01.006.

DYBÅ; MAIDEN; GLASS, 2014 — DYBÅ, T.; MAIDEN, N.; GLASS, R. 2014. The Reflective Software Engineer: Reflective Practice. IEEE Software. Vol 31. No 4. Pp. 32-36. DOI 10.1109/MS.2014.97

EIJNATTEN, 2003 — EIJNATTEN, F. M. 2003. Chaordic Systems Thinking: Chaos and complexity to explain human performance management. In G. D. Putnik, & A. Gunasekaran, Business Excellence 1: Performance measures, benchmarking and best practices in new economy (pp. 3-18). Portugal: University of Minho Press.

EISENHARDT, 1989 — EISENHARDT, K. 1989. Building Theories from Case Study Research. Academy of Management Review. Vol. 14. No. 4. pp. 532-550. DOI: 10.5465/AMR.1989.4308385

EISENHARDT; MARTIN, 2000 — EISENHARDT, K. M.; MARTIN, J. A. 2000. Dynamic Capabilities: What are they? Strategic Management Journal, 21, pp. 1105-1121.

FALESSI et al., 2014 — FALESSI, D.; FONTANA, R. M.; GIARDINO, C.; OLIVEIRA, R.; POWER K.; REJAB, M. M.; TAYLOR, K.; VALLON, R.; WANG, X. 2014. Trends and Emerging Areas of Agile Research: the report on XP2014 PhD Symposium. ACM SIGSOFT Software Engineering Notes. Vol. 39. No. 5. pp. 26-29. DOI: 10.1145/2659118.2659138

FONTANA; REINEHR; MALUCELLI, 2014 — FONTANA, R. M.; REINEHR, S.; MALUCELLI, A. 2014. Maturing in Agile: What Is It About? In: Proceedings of the 15th International Conference on Agile Software Development, XP 2014, Rome, Italy, May 26-30, pp. 94-109, DOI 10.1007/978-3-319-06862-6_7

FONTANA et al., 2014b — FONTANA, R. M.; FONTANA, I. M.; GARBUIO, P. A. R.; REINEHR, S.; MALUCELLI, A. 2014. Processes versus people: How should agile software development maturity be defined? The Journal of Systems and Software. Vol. 97. pp.140-155. DOI: 10.1016/j.jss.2014.07.030

FONTANA et al., 2015 — FONTANA, R. M.; MEYER Jr., V.; REINEHR, S.; MALUCELLI, A. 2015. Progressive outcomes: A framework for maturing in agile software development. The Journal of Systems and Software. Vol. 102. pp. 88-108. DOI: 10.1016/j.jss.2014.12.032

FONTANA et al., 2015b — FONTANA, R. M.; MEYER Jr., V.; REINEHR, S.; MALUCELLI, A. 2015. Management Ambidexterity: A Clue for Maturing in Agile Software Development. In: Casper Lassenius; Torgeir Dingsøyr; Maria Paasivaara. (Org.). Lecture Notes in Business Information Processing. 1ed. Switzerland: Springer International Publishing, 2015, v. 212, p. 199-204. DOI 10.1007/978-3-319-18612-2_17

FONTANA et al., 2015c — FONTANA, R. M.; REINEHR, S.; MALUCELLI, A. 2015. Progressive Outcomes: is it a handy approach to support agile methods process improvement? In: Proceedings of the XIV Simpósio Brasileiro de Qualidade de Software. Manaus, Aug. 17-21. pp. 94-106.

FONTANA et al., 2015d — FONTANA, R. M.; REINEHR, S.; MALUCELLI, A. 2015. Agile Compass: A Tool for Identifying Maturity in Agile Software-Development Teams. IEEE Software. Vol. 32. No. 6. pp. 20-23. DOI 10.1109/MS.2015.135

FORZA, 2002 — FORZA, C. 2002. Survey research in operations management: a process-based perspective. Int J Oper Prod Man. Vol. 22. No. 2, pp 152-194. DOI 10.1108/01443570210414310.

GANDOMANI et al., 2014 — GANDOMANI, T. J.; ZULZALIL, H.; GHANI, A. A. A.; SULTAN, A. B. Md.; PARIZI, R. M. 2014 The impact of inadequate and dysfunctional training on Agile transformation process: A Grounded Theory study, Inform. Softw. Technol. DOI: 10.1016/j.infsof.2014.05.011

GANDOMANI; NAFCHI, 2015	GANDOMANI, T. J.; NAFCHI, M. Z. 2015. An empirically-developed framework for Agile transition and adoption: A Grounded Theory approach. The Journal of Systems and Software. vol. 107. pp. 204-219. DOI: 10.1016/j.jss.2015.06.006
GIBSON; BIRKINSHAW, 2004	GIBSON, C., BIRKINSHAW, J. 2004. The antecedents, consequences, and mediating role of organizational ambidexterity. Academy of Management Journal, Vol. 47, No. 2, pp. 209-226.DOI: 10.2307/20159573
GREN; TORKAR; FELDT, 2015	GREN, L.; TORKAR, R.; FELDT, R. 2015. The prospects of a quantitative measurement of agility: A validation study on an agile maturity model. The Journal of Systems and Software. vol. 107. pp. 38-49. DOI: 10.1016/j.jss.2015.05.008
GÜTTEL; KONLECHNER, 2009	GÜTTEL, W. H.; KONLECHNER, S. W. 2009. Continuously Hanging By a Thread: Managing Contextually Ambidextrous Organizations. Schmalenbach Business Review. Vol. 61. pp. 150-171. Available at SSRN: http://ssrn.com/abstract=1406948
HAIR et al., 2006	HAIR, J.; BLACK, B.; BABIN, B.; ANDERSON, R. E.; TATHAN, R. L., 2006. Multivariate Data Analysis. Sixth Edition. Prentice Hall.
HARRIS; COLLINS; HEVNER, 2009	HARRIS, M. L.; COLLINS, R. W.; HEVNER, A. R. 2009. Control of Flexible Software Development Under Uncertainty. Information Systems Research. Vol. 20. No.3. pp. 400-419. DOI: 10.1287/isre.1090.0240.
HAZZAN; LERON, 2010	HAZZAN, O.; LERON, U. 2010. Disciplined and free-spirited: 'Time-out behaviour' at the Agile conference. The Journal of Systems and Software. Vol. 83. No. 11. 2010, pp. 2363-2365. DOI: 10.1016/j.jss.2010.06.018
HEVNER et al., 2004	HEVNER, A. R; MARCH, S. T.; PARK, J.; RAM, S. 2004. Design Science in Information Systems Research. MIS Quarterly. Vol. 28. No. 1. pp. 75-105.
HIDALGO, 2011	HIDALGO, C. 2011. The Value in Between: Organizations as Adapting and Evolving Networks. In P. Allen, S. Maguire, & B. McKelvey, The SAGE Handbook of Complexity and Management. London: SAGE Publications.
HODA; NOBLE; MARSHALL, 2012	HODA, R., NOBLE, J., MARSHALL, S. 2012. Self-Organizing Roles on Agile Software Development Teams. IEEE Transactions on Software Engineering, PP, p. 1. doi:10.1109/TSE.2012.30
HODA; NOBLE; MARSHALL, 2011	HODA, R.; NOBLE, J.; MARSHALL, S. 2011. The impact of inadequate customer collaboration on self-organizing Agile teams. Information and Software Technology. Vol. 53. No. 5. pp. 521–534. DOI: 10.1016/j.infsof.2010.10.009

HUMPHREY et al., 2010	HUMPHREY, W., CHICK, T., NICHOLS, W., POMEROY-HUFF, M. 2010. Team Software Process (TSP) Body of Knowledge (BOK), Software Engineering Institute, Carnegie Mellon University, Pittsburgh, Pennsylvania, Technical Report CMU/SEI-2010-TR-020. Available at http://goo.gl/x8frdu.
ISO/IEC, 2004	ISO/IEC: 15504-1. 2004. Information technology - Process assessment - Part 1: Concepts and vocabulary. ISO/IEC, Geneva, Switzerland. Available at http://goo.gl/DZJfuS
ISO/IEC, 2008	ISO/IEC: 12207. 2008. Systems and software engineering - Software life cycle processes, ISO/IEC, Geneva, Switzerland. Available at http://goo.gl/nRYgrI
JAKOBSEN; JOHNSON, 2008	JAKOBSEN, C. R.; JOHNSON, K. A. 2008. Mature Agile with a Twist of CMMI. In: Proceedings of the Agile Conference 2008. 4-8 Aug. pp. 212-217. DOI 10.1109/Agile.2008.10.
JIANG et al., 2004	JIANG, J. J.; KLEIN, G.; HWANG, H.; HUANG, J.; HUNG, S.. An exploration of the relationship between software development process maturity and project performance. Information & Management. 41. pp. 279-288. 2004. DOI: 10.1016/S0378-7206(03)00052-1
JOHNSON; WICHERN, 2007	JOHNSON, Richard A.; WICHERN, Dean W. 2007. Applied Multivariate Statistical Analysis. 6th ed. New Jersey: Person Prentice Hall.
KETTUNEN, 2012	KETTUNEN, P. 2012. Systematizing Software Development Agility: Towards an Enterprise Capability Improvement Framework. J Enterp Transform, Vol. 2, No. 2. pp. 81-104. DOI 10.1080/19488289.2012.664610.
KETTUNEN, 2014	KETTUNEN, P. 2014. Realizing Agile Software Enterprise Transformations by Team Performance Development. In: Cantone, G.; Marchesi, M. (eds.): XP 2014. LNBIP 179. pp. 285-293. DOI: 10.1007/978-3-319-06862-6_22
KIRK; TEMPERO, 2012	KIRK, D; TEMPERO, E. 2012. A lightweight framework for describing software practices. The Journal of Systems and Software, Vol. 85, No. 3, pp. 582-595. DOI: 10.1016/j.jss.2011.09.024
KITCHENHAM et al., 2002	KITCHENHAM, B. A.; PFLEEGER, S. L.; PICKARD, L. M.; JONES, P. W.; HOAGLIN, D. C.; EMAM, K. E.; ROSENBERG, J. 2002. Preliminary Guidelines for Empirical Research in Software Engineering. IEEE T Softw Eng. Vol. 28. No. 8. Aug. pp. 721-734. DOI 10.1109/TSE.2002.1027796.
KOHLEGGER; MAIER; THALMANN, 2009	KOHLEGGER, M.; MAIER, R.; THALMANN, S. 2009. Understanding Maturity Models Results of a Structured Content Analysis. Proceedings of the I-KNOW ’09 and I-SEMANTICS ’09. 2-4 September 2009. Available at http://goo.gl/hnw7uh

KRISHNA; BASU, 2012 — KRISHNA, V.; BASU, A. 2012. Minimizing technical debt: developer's viewpoint. Proceedings of the International Conference on Software Engineering and Mobile Application Modelling and Development (ICSEMA 2012).Chennai, India. 19-21 Dec. pp. 14-18. DOI: 10.1049/ic.2012.0147

KUIPERS; STOKER, 2009 — KUIPERS, B.S., STOKER, J.I. 2009. Development and performance of self-managing work teams: a theoretical and empirical examination. Int. J. Hum. Resour. Manage, Vol. 20, No. 2, pp. 399–419. doi:10.1080/09585190802670797

KURAPATI; MANYAM; PETERSEN, 2012 — KURAPATI, N.; MANYAM, V. S.; PETERSEN, K. 2012. Agile Software Development Practice Adoption Survey. In: Wohlin, C. Proceedings of the 13th International Conference on Agile Software Development, XP 2012, Malmö, Sweden, May 21-25. pp. 16-30. DOI: 10.1007/978-3-642-30350-0_2

LAYMAN; WILLIAMS; CUNNINGHAM, 2004 — LAYMAN, L., WILLIAMS, L., CUNNINGHAM, L. 2004. Motivations and Measurements in an Agile Case Study. Proceedings of the 2004 Workshop on Quantitative techniques for software agile process, pp. 14-24, DOI: 10.1145/1151433.1151436

LEE; MOHAREJI, 2012 — LEE, A. S.; MOHAREJI, K. 2012. Linking Relevance to Practical Significance. Proceedings of the 45th Hawaii International Conference on System Sciences. 4-7 Jan. Maui, HI. pp. 5234-5240. DOI 10.1109/HICSS.2012.416

LEPPÄNEN, 2013 — LEPPÄNEN, M. 2013. A Comparative Analysis of Agile Maturity Models. In: R. Pooley et al. (eds.), Information Systems Development: Reflections, Challenges and New Directions. Springer Science+Business Media. New York. pp 329-343. DOI: 10.1007/978-1-4614-1951-5_27

LINA; DAN, 2012 — LINA, Z.; DAN, S. 2012. Research on Combining Scrum with CMMI in Small and Medium Organizations. International Conference on Computer Science and Electronics Engineering. Hangzhou, 23-25 March, pp. 554-557. DOI: 10.1109/ICCSEE.2012.477

LOHAN; LANG; CONBOY, 2011 — LOHAN, G.; LANG, M.; CONBOY, K.. 2011. Having a Customer Focus in Agile Software Development. In.: Pokorny et al. (eds). Information Systems Development: Business Systems and Services: Modeling and Development. pp 441-453. DOI: 10.1007/978-1-4419-9790-6_35

LUI; CHAN, 2005 — LUI, K. M.; CHAN, K. C. C. 2005. A Road Map for Implementing eXtreme Programming. In: M.Li, B. Boehm, and L. J. Osterweil (eds.): SPW 2005, LNCS 3840, pp. 474-481. DOI 10.1007/11608035_38.

LUKASIEWICZ; MILER, 2012 — LUKASIEWICZ, K.; MILER, J., 2012. Improving agility and discipline of software development with the Scrum and CMMI.

IET Software, Vol 6, No. 5, pp. 416-422. DOI: 10.1049/iet-sen.2011.0193.

MAGUIRE; ALLEN; MCKELVEY, 2011 — MAGUIRE, S., ALLEN, P., MCKELVEY, B. 2011. Complexity and Management: Introducing SAGE Handbook. In: S. Maguire, P. Allen, & B. McKelvey, The SAGE Handbook of Complexity and Management. pp. 1-26. Thousand Oaks: SAGE Publications.

MAIER; MOUTRIE; CLARKSON, 2012 — MAIER, A. M.; MOUTRIE, J.; CLARKSON, J. 2012. Assessing Organizational Capabilities: Reviewing and Guiding the Development of Maturity Grids. IEEE Transactions on Engineering Management, Vol. 59, No. 1, Feb. pp. 138-159. DOI 10.1109/TEM.2010.2077289.

MANEN; VLIET, 2014 — MANEN, H.; VLIET, H. 2014. Organization-Wide Agile Expansion Requires an Organization-Wide Agile Mindset. In: Jedlitschka, A. et al. (orgs). 15th International Conference, PROFES 2014, Helsinki, Finland, December 10-12, 2014. Proceedings. Pp.48-62. DOI: 10.1007/978-3-319-13835-0_4

MARCH, 1988 — MARCH, J. 1988. The technology of foolishness. In: MARCH, J. (ed.) Decisions and organizations. Oxford: Blackwell, pp. 253-265.

MARCH, 1991 — MARCH, J. G. 1991. Exploration and Exploitation in Organizational Learning. Organization Science, 2.

MCDANIEL JR., 2007 — MCDANIEL JR., R. R. 2007. Management Strategies for Complex Adaptive Systems. Performance Improvement Quarterly, Vol. 20, No. 2, pp. 21-42. DOI: 10.1111/j.1937-8327.2007.tb00438.x

MCHUGH, CONBOY, LANG, 2012 — MCHUGH, O.; CONBOY, K.; LANG, M. 2012. Agile Practices: The Impact on Trust in Software Project Teams. IEEE Software, Vol. 29, No. 3, pp. 71-76, DOI: 10.1109/MS.2011.118

MELO et al., 2013 — MELO, C. O., SANTOS, V., KATAYAMA, E., CORBUCCI, H., PRIKLADNICKI, R., GOLDMAN, A., Kon, F. 2013. The evolution of agile software development in Brazil. Journal of the Brazilian Computer Society, Vol. 19, No. 4, pp. 523-552, DOI: 10.1007/s13173-013-0114-x. Available at http://goo.gl/843M3O

MELO et al., 2013b — MELO, C. O.; CRUZES, D.; KON, Fábio; CONRADI, Reidar. 2013. Interpretative case studies on agile team productivity and management. Information and Software Technology. Vol. 55. No. 2. pp. 412-427. DOI: 10.1016/j.infsof.2012.09.004

MIDDLETON; JOYCE, 2012 — MIDDLETON, P.; JOYCE, D. 2012. Lean Software Management: BBC Worldwide Case Study. IEEE Transactions on Engineering Management, Vol. 59, No. 1, pp. 20-32. DOI: 10.1109/TEM.2010.2081675

MISRA; KUMAR; KUMAR, 2009 — MISRA, S. C.; KUMAR, V.; KUMAR, U. 2009. Identifying some important success factors in adopting agile software development practices. Journal of Systems and Software, Vol. 82, No. 11, pp. 1869-1890, DOI: 10.1016/j.jss.2009.05.052

MITLETON-KELLY, 2003 — MITLETON-KELLY, E. 2003. Ten Principles of Complexity & Enabling Infrastructures. In: E. Mitleton-Kelly, Complex Systems and Evolutionary Perspectives of Organisations: the application of complexity theory to organizations. Oxford: Elsevier Science Ltd.

MOE; AURUM; DYBÅ, 2012 — MOE, N. B.; AURUM, A., DYBÅ, T. 2012. Challenges of shared decision-making: A multiple case study of agile software development. Information and Software Technology, Vol. 54, No. 8, pp. 853-865, DOI: 10.1016/j.infsof.2011.11.006

MOE; DINGSØYR; DYBÅ, 2009 — MOE, N. B.; DINGSØYR, T.; DYBÅ, T. 2009. Overcoming barriers to self-management in software teams. IEEE Software, Vol. 26, No. 6, pp. 20-26, DOI 10.1109/MS.2009.182

MOE; DINGSØYR; DYBÅ, 2010 — MOE, N. B.; DINGSØYR, T.; DYBÅ, T. 2010. A teamwork model for understanding an agile team: A case study of a Scrum project. Information and Software Technology, Vol 52, No. 5, pp. 480-491, DOI: 10.1016/j.infsof.2009.11.004

NAWROCKI; WALTER; WOJCIECHOWSKI, 2001 — NAWROCKI, J.; WALTER, B.; WOJCIECHOWSKI, A. 2001. Toward Maturity Model for eXtreme Programming. In: Proceedings of the 17th Euromicro Conference., 04-06 Sep, Warsaw, pp. 233-239, DOI 10.1109/EURMIC.2001.952459.

NICOLAI; SEIDL, 2010 — NICOLAI, A.; SEIDL, D.. 2010. That's Relevant! Different Forms of Practical Relevance in Management Science. Organization Studies, Vol. 31, No. 9-10, pp. 1257-1285, DOI 10.1177/0170840610374401.

OLSSON; BOSCH, 2014 — OLSSON, H. H.; BOSCH, J. 2014. Towards Agile and Beyond: An Empirical Account on the Challenges Involved When Advancing Software Development Practices. In: Cantone, G.; Marchesi, M. (eds.): XP 2014, LNBIP 179, pp. 327-335. DOI: 10.1007/978-3-319-06862-6_27

OUCHI, 1979 — OUCHI, W. G. 1979. A Conceptual Framework for the Design of Organizational Control Mechanisms. Management Science. Vol. 25. No. 9. pp. 833-848. DOI: 10.1287/mnsc.25.9.833

ÖZCAN-TOP; DEMIRÖRS, 2013 — ÖZCAN-TOP, Ö.; DEMIRÖRS, O. 2013. Assessment of Agile Maturity Models: A Multiple Case Study. In: Software Process Improvement and Capability Determination, 13th International Conference, SPICE 2013, Bremen, Germany, June 4-6. Proceedings. pp 130-141. DOI: 10.1007/978-3-642-38833-0_12. 2013.

ÖZCAN-TOP; DEMIRÖRS, 2014 — ÖZCAN-TOP, Ö; DEMIRÖRS, O. 2014. Assessing Software Agility: An Exploratory Case Study. In: Mitasiunas et al. (Orgs), Software Process Improvement and Capability Determination:

14th International Conference, SPICE 2014, Vilnius, Lithuania, November 4-6, 2014, Proceedings. pp. 202-213. DOI 10.1007/978-3-319-13036-1_18

PAASIVAARA; LASSENIUS, 2014 — PAASIVAARA, M.; LASSENIUS, C. 2014. Communities of practice in large distributed agile software development organization – Case Ericsson. Information and Software Technology, Vol. 56, No. 12, pp. 1556-1577, DOI: 10.1016/j.infsof.2014.06.008

PAASIVAARA et al., 2014 — PAASIVAARA, M.; VÄÄTTÄNEN, O., HALLIKAINEN, M.; LASSENIUS, C. Supporting a Large-Scale Lean and Agile Transformation by Defining Common Values. 2014. In: Dingsoyr, T. et al. (orgs). XP 2014 International Workshops, Rome, Italy, May 26-30, 2014, Revised Selected Papers. pp. 73-82. DOI: 10.1007/978-3-319-14358-3_7

PACKLICK, 2007 — PACKLICK, J. 2007. The Agility Maturity Map – a Goal Oriented Approach to Agile Improvement. In: Proceedings of the Agile Conference 2007. 13-17 Aug, pp. 266-271, DOI 10.1109/AGILE.2007.55

PATEL; RAMACHANDRAN, 2009 — PATEL, C.; RAMACHANDRAN, M. 2009. Agile Maturity Model (AMM): A Software Process Improvement framework for Agile Software Development Practices. International Journal of Software Engineering, Vol. 2, No. 1, pp. 3-28. Available at http://goo.gl/FGe0eE

PAULK, 2001 — PAULK, M., 2001. Extreme Programming from a CMM Perspective. IEEE Software, Vol. 18, No. 6, pp. 19-26, DOI 10.1109/52.965798.

PERROW, 1981 — PERROW, C. 1981, July-August. Normal Accident at Three Mile Island. Society.

POPPENDIECK; POPPENDIECK, 2003 — POPPENDIECK, M.; POPPENDIECK, T. Lean Software Development: An Agile Toolkit. Addison Wesley, 2003.

POWER, 2014 — POWER, K. 2014. Social Contracts, Simple Rules and Self-organization: A Perspective on Agile Development. In: Cantone, G; Marchesi, M. (eds.) Lecture Notes in Business Information Processing. vol 179. pp. 277-284. DOI: 10.1007/978-3-319-06862-6_21

POWER; CONBOY, 2014 — POWER, K.; CONBOY, K. 2014. Impediments to Flow: Rethinking the Lean Concept of "Waste" in Modern Software Development. In: Cantone, G.; Marchesi, M. (eds.): XP 2014. LNBIP 179, pp. 2013-217. DOI: 10.1007/978-3-319-06862-6_14

QUMER; HENDERSON-SELLERS, 2008 — QUMER, A.; HENDERSON-SELLERS, B. 2008. A framework to support the evaluation, adoption and improvement of agile methods in practice. Journal of Systems and Software, Vol. 81, No. 11, pp. 1899-1919, DOI 10.1016/j.jss.2007.12.806

RAISH; BIRKINSHAW, 2008 — RAISH, S., BIRKINSHAW, J. 2008. Organizational Ambidexterity: Antecedents, Outcomes and Moderators. Journal of Management. Vol. 34, No. 3, pp. 375-409, DOI: 10.1177/0149206308316058

RAMESH; MOHAN; CAO, 2012 — RAMESH, B.; MOHAN, K.; CAO, L. 2012. Ambidexterity in Agile Distributed Development: An Empirical Investigation. Information Systems Research, Vol. 23, No. 2, pp. 323-339, DOI: 10.1287/isre.1110.0351

SANTOS; GOLDMAN; SOUZA, 2014 — SANTOS, V.; GOLDMAN, A.; SOUZA, C. R. B. S. 2014. Fostering effective inter-team knowledge sharing in agile software development. Empirical Software Engineering, Vol. 20, No. 4, pp. 1006-1051, DOI: 10.1007/s10664-014-9307-y

SCHWEIGERT et al., 2012 — SCHWEIGERT, T.; NEVALAINEN, R.; VOHWINKEL, D.; KORSAA, M.; BIRO, M., 2012. Agile Maturity Model: Oxymoron or the Next Level of Understanding. A. Mas. Et al. (Eds). : SPICE 2012, May 29-31, pp. 289-294. DOI 10.1007/978-3-642-30439-2_34.

SHARP; ROBINSON; PETRE, 2009 — SHARP, H.; ROBINSON, H.; PETRE, M. 2009. The role of physical artefacts in agile software development: Two complementary perspectives. Interacting with computers, Vol. 21, No. 1, pp. 108-116, DOI: 10.1016/j.intcom.2008.10.006

SHEFFIELD; LEMÉTAYER, 2012 — SHEFFIELD, J.; LEMÉTAYER, J. 2012. Factor associated with the software development agility of successful projects. International Journal of Project Management, Vol. 31, No. 3, pp. 459-472, DOI: http://dx.doi.org/10.1016/j.ijproman.2012.09.011

SIDKY; ARTHUR; BOHNER, 2007 — SIDKY, A.; ARTHUR, J.; BOHNER, S. 2007. A disciplined approach to adopting agile practices: the agile adoption framework. Innovation in Systems and Software Engineering, Vol. 3, No. 3, pp. 203-216, DOI: 10.1007/s11334-007-0026-z.

SILVA et al., 2015 — SILVA, F. S.; SOARES, F. S. F.; PERES, A. L.; AZEVEDO, I. M.; VASCONCELOS, A. P. L. F.; KAMEI, F. K.; MEIRA, S. R. L. 2015. Using CMMI together with agile software development: A systematic review. Information and Software Technology, Vol. 58, pp. 20-43, DOI: 10.1016/j.infsof.2014.09.012

SJØBERG et al., 2008 — SJØBERG, D. I. K.; DYBÅ, T.; ANDA, B. C. D.; HANNAY, J. E. 2008. Building Theories in Software Engineering. In: Shull, F. et al. (eds). Guide to Advanced Empirical Software Engineering. pp 312-336. DOI 10.1007/978-1-84800-044-5_12

SNOWDEN; BOONE, 2007 — SNOWDEN, D. J.; BOONE, M. E. 2007. A leader's framework for decision-making. Harvard Business Review, pp. 68–76

SOFTEX, 2012 — SOFTEX. 2012. Software e Serviços de TI: A Indústria Brasileira em Perspectiva. Year 2012. Vol. 2. Available in

http://publicacao.observatorio.softex.br/_publicacoes/index.php

SOFTEX, 2012b — SOFTEX. 2012. MPS.BR - Melhoria de Processo do Software Brasileiro – Guia Geral MPS de Software. December 2012. Available in http://www.softex.br/mpsbr/guias/

SOUNDARAJAN; ARTHUR; BALCI, 2012 — SOUDARAJAN, S., ARTHUR, J. D., BALCI, O. 2012. A Methodology for Assessing Agile Software Development Methods. In: Proceedings of the Agile Conference 2012, pp. 51-54, DOI: 10.1109/Agile.2012.24

SPOELSTRA; IACOB; VAN SINDEREN, 2011 — SPOELSTRA, W.; IACOB, M.; VAN SINDEREN, M. 2011. Software Reuse in Agile Development Organizations – A Conceptual Management Tool. Proceedings of the 2011 ACM Symposium on Applied Computing, pp. 315-322, DOI: 10.1145/1982185.1982255

STACEY, 1996 — STACEY, R. 1996. Complexity and Creativity in Organizations. San Francisco: Berret-Koehler Publishers.

STACEY; GRIFFIN; SHAW, 2000 — STACEY, R., GRIFFIN, D., SHAW, P. 2000. Complexity and Management: Fad or radical challenge to systems thinking? London: Routledge.

STAPLES et al., 2007 — STAPLES, M.; NIAZI, M.; JEFFERY, R.; ABRAHAMS, A.; BYATT, P.; MURPHY, R. 2007. An exploratory study of why organizations do not adopt CMMI. Journal of Systems and Software, Vol. 80, No. 6, pp. 883-895, DOI: 10.1016/j.jss.2006.09.008

SUBRAMANIAN; JIANG; KLEIN, 2007 — SUBRAMANIAN, G. H.; JIANG, J. J.; KLEIN, G. 2007. Software quality and IS project performance improvements from software development process maturity and IS implementation strategies. Journal of Systems and Software. Vol. 80, No. 4, pp. 616-627. DOI: 10.1016/j.jss.2006.06.014

SUOMINEN; MÄKINEN, 2013 — SUOMINEN, M.; MÄKINEN, T. 2013. On the applicability of capability models for small software organizations: does the use of standard processes lead to a better achievement of business goals? Software Quality Journal, Vol. 22, No. 4, pp. 579-591, DOI: 10.1007/s11219-013-9201-7. 2013

SUTHERLAND; JAKOBSEN; JOHNSON, 2007 — SUTHERLAND, J.; JAKOBSEN, C. R.; JOHNSON, K., 2007. Scrum and CMMI Level 5: The Magic Potion for Code Warriors. In: Proceedings of the Agile Conference 2007, 13-17 Aug. pp. 272-278. DOI 10.1109/AGILE.2007.52.

TIWANA, 2008 — TIWANA, A. 2008. Do bridging ties complement strong ties? An empirical examination of alliance ambidexterity. Strategic Management Journal, Vol. 29, pp. 251-272, DOI: 10.1002/smj.666.

TIWANA, 2010 — TIWANA, A. 2010. Systems Development Ambidexterity: Explaining the Complementary and Substitutive Roles of Formal and Informal Controls. Journal of Management

Information Systems. Vol. 27. No. 2. pp. 87-126. DOI: 10.2753/MIS0742-1222270203.

TSOUKAS, 2005 — TSOUKAS, H. 2005. Complex knowledge: studies in organizational epistemology. New York: Oxford University Press.

TSOUKAS; HATCH, 2005 — TSOUKAS, H.; HATCH, M. J. Complex Thinking, Complex Practice: The Case for a Narrative Approach to Organizational Complexity. In: Tsoukas, H. Complex Knowledge: Studies in Organizational Epistemology. Oxford: Oxford University Press, 2005.

TUCKMAN, 1965 — TUCKMAN, B. W., 1965. Developmental sequence in small groups. Psychol. Bul, Vol. 63, No. 6, pp. 384–399 DOI: http://dx.doi.org/10.1037/h0022100.

TURNER; SWART; MAYLOR, 2013 — TURNER, N.; SWART, J.; MAYLOR, H. 2013. Mechanisms for Managing Ambidexterity: A Review and Research Agenda. International Journal of Management Reviews, Vol. 15, No. 3, pp. 317-332. DOI: 10.1111/j.1468-2370.2012.00343.x

TURNER; MAYLOR; SWART, 2015 — TURNER, N.; MAYLOR, H.; SWART, J. 2015. Ambidexterity in projects: An intellectual capital perspective. International Journal of Project Management. Vol. 33. No. 1. pp. 177-188. DOI: 10.1016/j.ijproman.2014.05.002. 2014.

TUSHMAN; O'REILLY III, 1996 — TUSHMAN, M. L.; O'REILLY III, C. A. 1996. Ambidextrous organizations: Managing evolutionary and revolutionary change. California Management Review, Vol. 38, No. 4, pp. 8-30.

VENABLE; PRIES-HEJE; BASKERVILLE, 2012 — VENABLE, J.; PRIES-HEJE, J.; BASKERVILLE, R. 2012. A Comprehensive Framework for Evaluation in Design Science Research. In: Peffers, K.; Rothenberger, M; Kuechler, B. (Eds.): DESRIST 2012, LNCS 7286, pp. 423-438, DOI: 10.1007/978-3-642-29863-9_31

VERSION ONE, 2015 — VERSION ONE. 2015. 9th Annual State of Agile Survey. Available at http://www.versionone.com/pdf/state-of-agile-development-survey-ninth.pdf. Accessed in Aug, 2015.

VIDGEN; WANG, 2009 — VIDGEN, R.; WANG, X. 2009. Coevolving Systems and the Organization of Agile Software Development. Information Systems Research, Vol. 20, n. 3, pp. 355–376, DOI 10.1287/isre.1090.0237

VINEKAR; SLINKMAN; NERUR, 2006 — VINEJAR, V.; SLINKMAN, C. W.; NERUR, S. 2006. Can agile and traditional systems development approaches coexist? An ambidextrous view. Information Systems Management, Vol. 23, No. 3, pp. 31-42, DOI: 10.1201/1078.10580530/46108. 23.3.20060601/93705.4

WAARDENBURG; VLIET, 2013	WAARDENBURG, G.; VLIET, H. 2013. When agile meets the enterprise. Information and Software Technology, Vol. 55, No. 12, pp. 2154-2171, DOI: 10.1016/j.infsof.2013.07.012
WALTER et al., 2015	WALTER, M.; TRAMONTINI, R.; FONTANA, R. M.; REINEHR, S.; MALUCELLI, A. From Sprints to Lean Flow: Management Strategies for Agile Improvement. In: Casper Lassenius; Torgeir Dingsøyr; Maria Paasivaara. (Org.). Lecture Notes in Business Information Processing. 1ed. Switzerland: Springer International Publishing, 2015, v. 212, p. 310-318. DOI 10.1007/978-3-319-18612-2_32
WANG; CONBOY; CAWLEY, 2012	WANG, X.; CONBOY, K.; CAWLEY, O. 2012. “Leagile” software development: An experience report analysis of the application of lean approaches in agile software development. Journal of Systems and Software, Vol. 85, No. 6, pp. 1287-1299, DOI: 10.1016/j.jss.2012.01.061
WEICK; SUTCLIFFE; OBSTFELD, 2005	WEICK, K. E.; SUTCLIFFE, K. M.; OBSTFELD, D. 2005. Organizing and the Process of Sensemaking. Organization Science, Vol. 26, No. 14, pp. 409-421, DOI: 10.1287/ orsc.1050.0133
WHITWORTH; BIDDLE, 2007	WHITWORTH, E.; BIDDLE, R. 2007. The Social Nature of Agile Teams. In: Proceedings of the Agile Conference 2007, pp. 26-36, DOI: 10.1109/AGILE.2007.60
WILLIAMS et al., 2004	WILLIAMS, L., KREBS, W., LAYMAN, L., ANTÓN, A. 2004. Toward a Framework for Evaluating Extreme Programming. In: Proceedings of the 8th International Conference on Empirical Assessment in Software Engineering, pp. 11-20.
WILLIAMS; RUBIN; COHN, 2010	WILLIAMS, L., RUBIN, K., COHN, M. 2010. Driving Process Improvement Via Comparative Agility Assessment. In: Proceedings of the Agile Conference, pp. 3-10, 10.1109/AGILE.2010.12
WNUK; GORSCHEK; ZAHDA, 2013	WNUK, K.; GORSCHEK, T.; ZAHDA, S. 2013. Obsolete software requirements. Information and Software Technology, Vol. 55, No. 6, pp. 921–940, DOI: 10.1016/j.infsof.2012.12.001
WOOD; MICHAELIDES; THOMSON, 2012	WOOD, S.; MICHAELIDES, G.; THOMSON, C. 2012. Successful extreme programming: Fidelity to the methodology or good teamworking?, Information and Software Technology, Vol. 55, No. 4, pp. 660-672, DOI: 10.1016/j.infsof.2012.10.002
YIN; FIGUEIREDO; SILVA, 2011	YIN, A.; FIGUEIREDO, S.; SILVA, M. M. 2011. Scrum Maturity Model: Validation for IT organizations’ roadmap to develop software centered on the client role. In: Proceedings of the ICSEA 2011, The Sixth International Conference on Software Engineering Advances, pp 20-29, 23-29 Oct, Barcelona. Available at http://goo.gl/SklUZr
YIN, 2005	YIN, Robert K. Estudo de Caso: Planejamento e Métodos. 3ª Ed. Porto Alegre: Bookman, 2005

APPENDIX A – QUESTIONNAIRE STATEMENTS FOR STAGE 1 SURVEY

Author	Number	Related survey statement
Soundararajan et al. (2012)	X2	Allowing requirements to evolve during the project
	X13	Doing code refactoring
	X14	Doing pair programming
	X47	Doing things when they have to be done, not before
	X48	Self-organizing
	X60	Distributing physically so as to reflect agile philosophy
	X30	Delivering working software continuously
	X49	Giving continuous feedback
	X1	Using product backlog to define requirements
	X50	Writing agile documentation
	X31	Making agile project estimates
	X36	Holding retrospective meetings
	X68	Allowing customer to drive iterations
	X51	Distributing expertise on the team appropriately
	X37	Tracking and reporting iteration progress
Buglione (2011)	X43	Collaborating with team members
	X38	Integrating management activities directly into development tasks
	X44	Keeping work simple
	X23	Managing source code
	X67	Developing products that respond to business needs
	X3	Using stories to define requirements
	X46	Sharing responsibility
	X22	Managing software configuration (version control)
	X57	Being geographically distributed (different cities or countries)
	X59	Dealing easily with regulatory compliance
	X63	Dealing easily with domain complexity
	X64	Dealing easily with technical complexity
	X58	Dealing easily with organizational complexity
	X65	Dealing easily with enterprise discipline
Abbas et al. (2010)	X8	Performing traditional systems analysis
	X16	Being concerned about database architecture
	X54	Doing agile quality assurance
	X41	Communicating face-to-face daily
	X53	Analyzing and inspecting code
	X52	Running lightweight tests and reviews
	X11	Specifying software architecture
	X15	Using code standards
	X7	Defining lightweight requirements
	X28	Doing iterative and incremental development
	X66	Having the customer actively participate during the project
Williams et al. (2010)	X61	Being multidisciplinary
	X40	Focusing on work (priorities do not change during iteration)
	X6	Eliciting requirements based on communication
	X5	Allowing the emergence of requirements
	X4	Doing technical design of requirements
	X25	Planning releases
	X32	Defining scope according to schedule

Author	Number	Related survey statement
	X35	Holding daily progress tracking meetings
	X33	Making estimates with the people who will do the work
	X26	Planning before and during the project
	X20	Doing test-driven development
	X17	Doing continuous code integration
	X12	Using collective code ownership
	X45	Not losing autonomy when under pressure to meet deadlines
	X39	Responding to pressure by re-prioritizing or re-scoping instead of working overtime or adding people
	X62	Encouraging a culture of working together as a team rather than individually
	X55	Implementing development infrastructure that supports agility
	X56	Developing people's agility skills
	X27	Using timeboxes in planning
	X42	Questioning and learning from one another
Layman et al. (2004)	X29	Making short software releases
	X69	Customer being collocated
	X24	Using the planning game
	X21	Collecting test metrics
	X18	Running automated unit tests
	X19	Running user acceptance tests
	X10	Using simple software design
	X34	Maintaining a sustainable pace (do minimum overtime)
	X9	Using metaphors to describe requirements
CMMI-DEV (CMMI Product Team, 2010)	X71	Analyzing possible decisions using a formal evaluation process that evaluates identified alternatives against established criteria
	X72	Managing projects according to an integrated and defined process
	X73	Collecting metrics that are used to support management information needs
	X74	Having defined process assets, work environment standards, and rules and guidelines
	X75	Getting to know strengths and weaknesses and be able to plan and implement process improvements based on that
	X76	Identifying gaps in performance and selecting and deploying improvements to close these gaps
	X77	Having a quantitative (metrics-based) understanding of processes
	X78	Developing people's skills and knowledge so they can perform their roles effectively and efficiently
	X79	Establishing and maintaining plans that define project activities
	X80	Evaluating processes and work products objectively and addressing non-compliance issues
	X81	Managing projects with measures and analytic techniques
	X82	Formally eliciting, analyzing, and validating requirements for the product and stakeholders
	X83	Planning and invoking risk handling activities as needed across the life of the project
	X84	Managing the acquisition of products and services from suppliers
	X85	Maintaining alignment between requirements and the project's plans and work products

APPENDIX B – QUESTIONNAIRE STATEMENTS FOR STAGE 2 – PHASE 1 SURVEY

1. Do you believe a model that aids teams to become mature in agile software development would be useful? ( ) Yes ( ) No. Why?
2. Based on your personal opinion, what would the road map to mature in agile methods be like? Please, number the practices bellow in a maturity growing sequence. Please add any comments you find necessary.

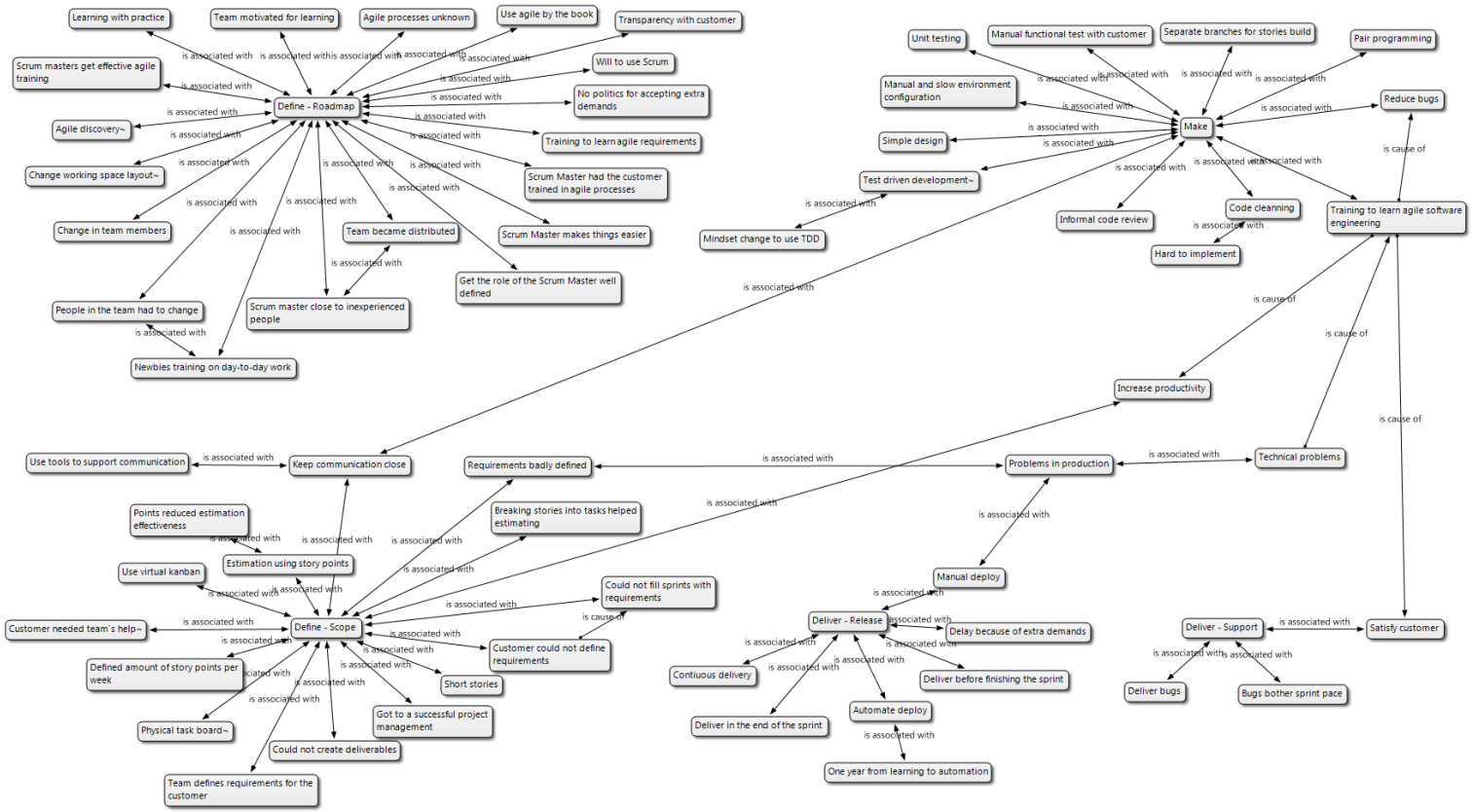
   _____ Focus on agile requirements;
   _____ Focus on software architecture;
   _____ Focus on agile coding;
   _____ Focus on agile testing;
   _____ Focus on agile planning;
   _____ Focus on agile project monitoring;
   _____ Focus on agile values in the team;
   _____ Focus on agile quality assurance;
   _____ Focus on defining an agile physical environment;
   _____ Focus on involved customer;
   _____ Focus on metrics;
   _____ Focus on defining processes;
   _____ Focus on controlling processes;
   _____ - _____;

APPENDIX C – QUESTIONNAIRE STATEMENTS FOR AMBIDEXTERITY EVALUATION IN CASE STUDY

Reference Author	Aspect	Question
Gibson & Birkinshaw (2004)	Performance	This team is achieving its full potential
		People at my level are satisfied with the level of team performance
		This team does a good job of satisfying our customers
		This team gives me the opportunity and encouragement to do the best work I am capable of
	Good Alignment	The management systems in this organization work coherently to support the overall objectives of this organization
	Bad Alignment	The management systems in this organization cause us to waste resources on unproductive activities
		People in this organization often end up working at cross-purposes because our management systems give them conflicting objectives
	Adaptability	The management systems in this organization encourage people to challenge outmoded traditions/practices/sacred cows
		The management systems in this organization is flexible enough to allow us to respond quickly to changes in our markets
		The management system in this organization evolves rapidly in response to shifts in our business priorities
Tiwana (2008)	Bridging Ties	Members of this team vary widely in their areas of expertise
		Members of this team have a variety of different backgrounds and experiences
		Members of this team have skills and abilities that complement those of one another
	Strong Ties	There is close, personal interaction among team members at multiple levels
		This project team is characterized by high reciprocity among members
		This project team is characterized by mutual trust among members
		This project team is characterized by mutual respect among members
		This project team is characterized by personal friendship among members

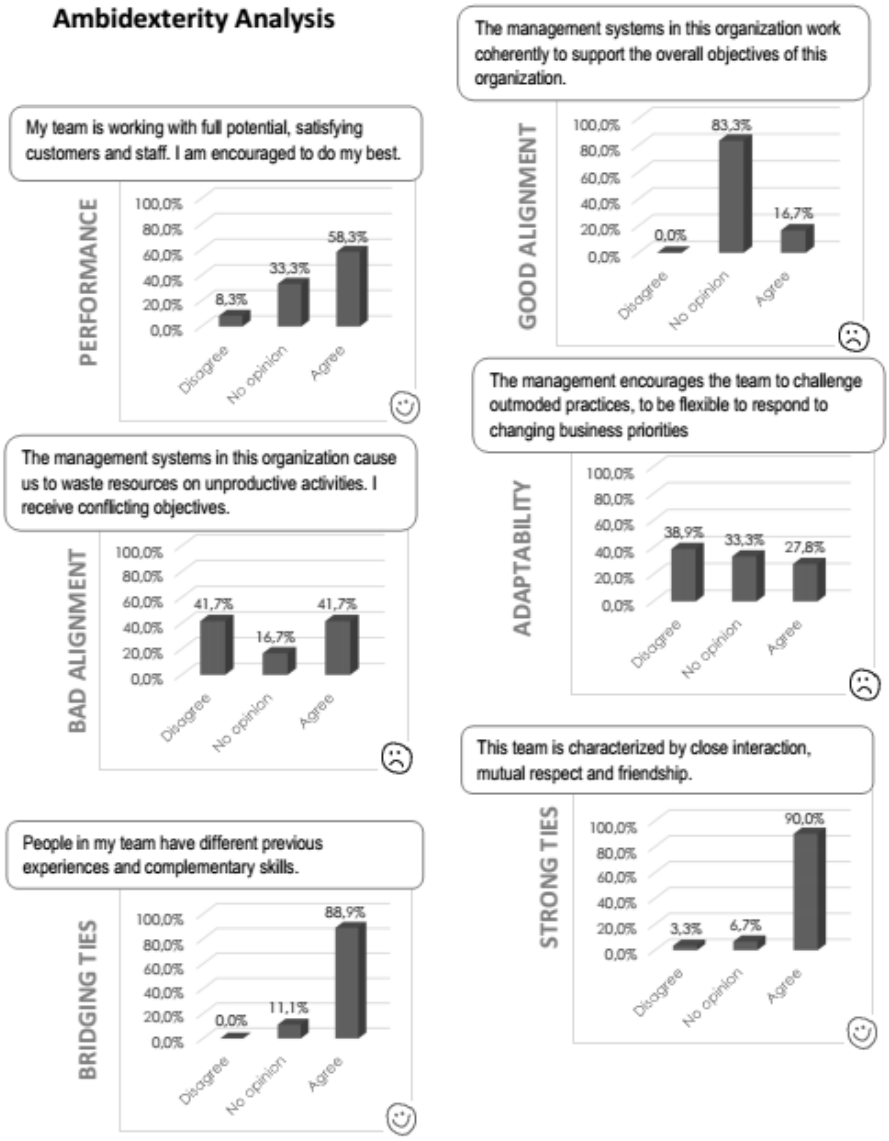
APPENDIX D – CODE NETWORK EXAMPLE

Code network for Company D – Past

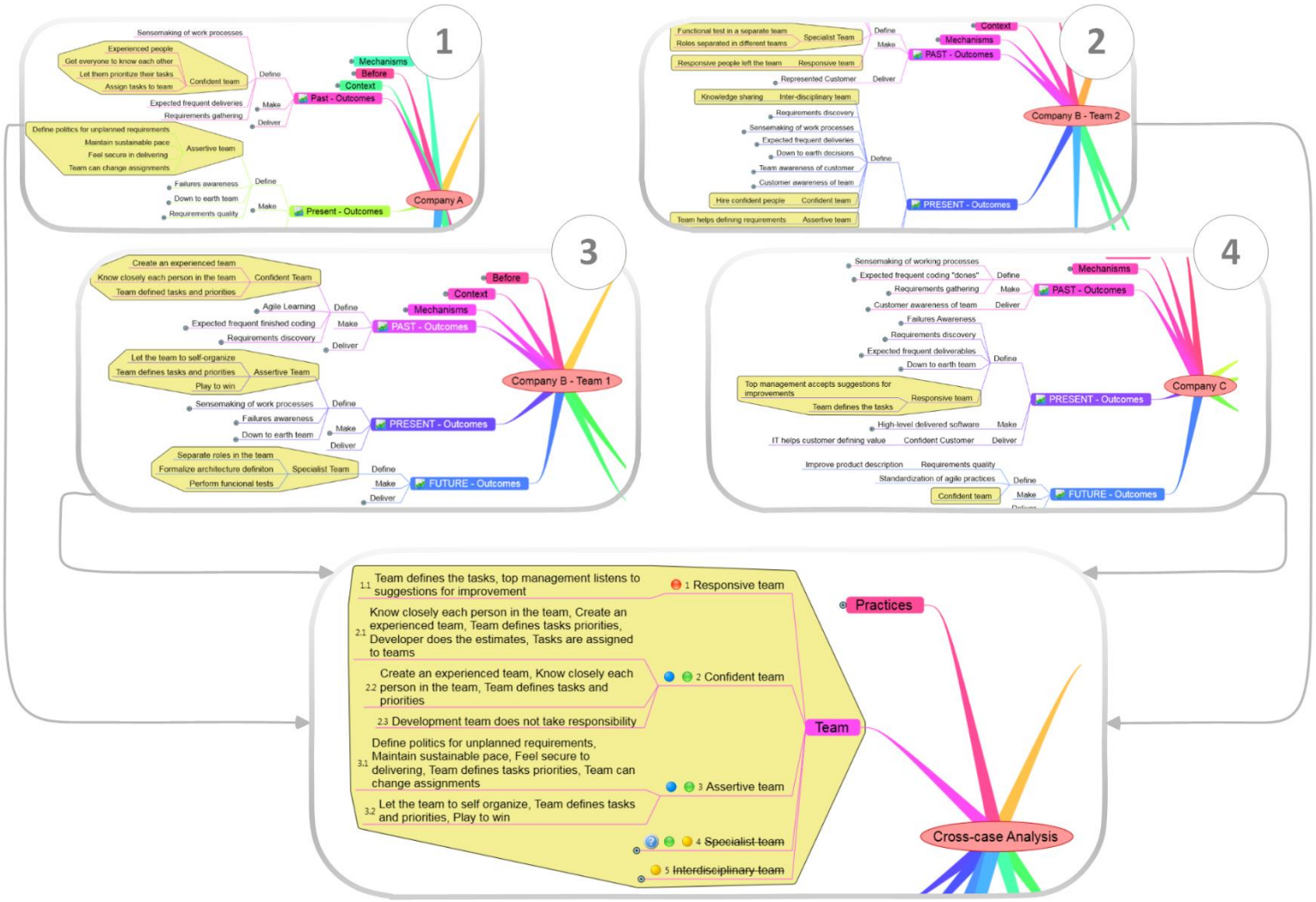


APPENDIX E – CASE REPORT AMBIDEXTERITY ANALYSIS

Ambidexterity analysis presented in Company C



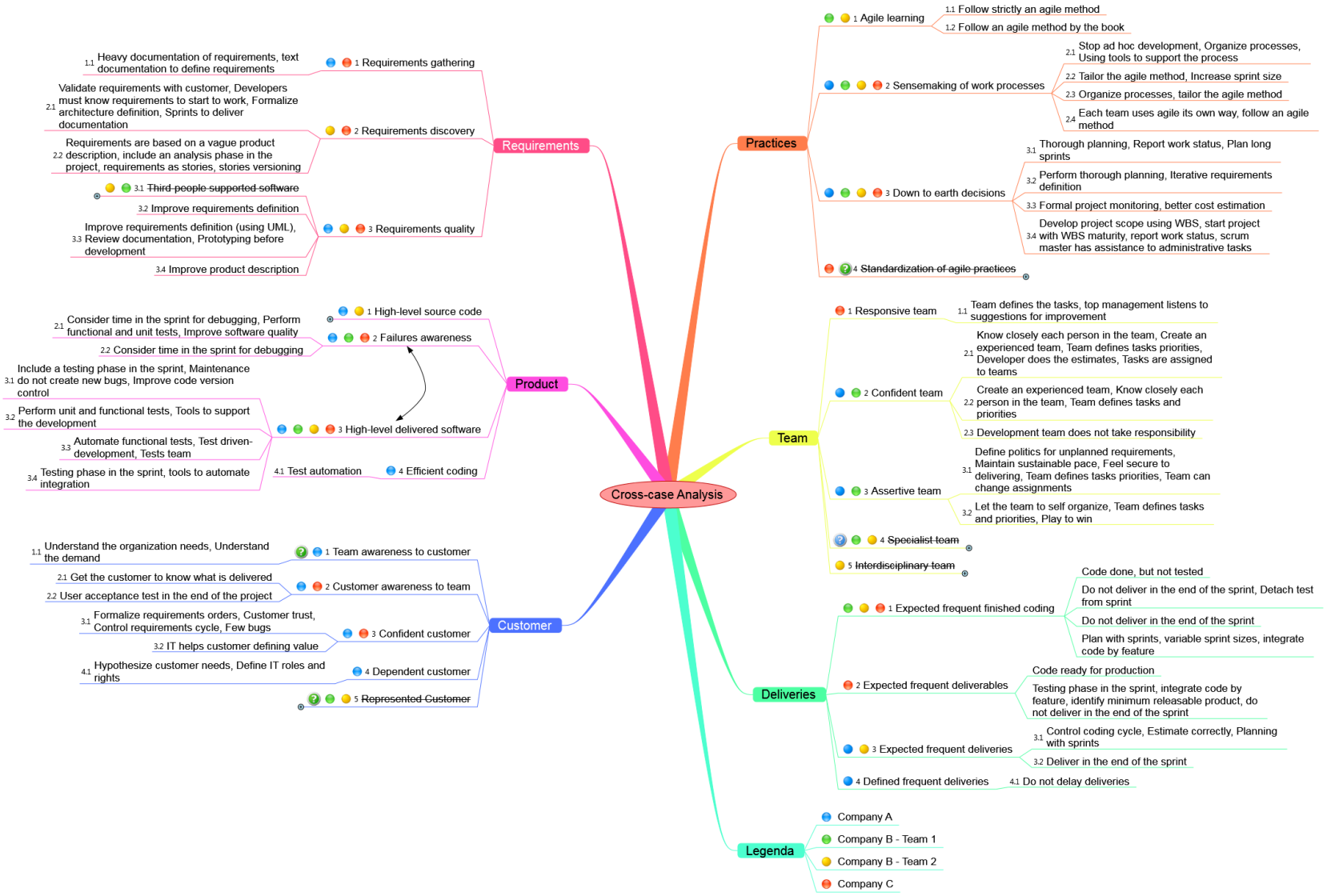
APPENDIX F – CROSS CASE ANALYSIS PROCEDURE



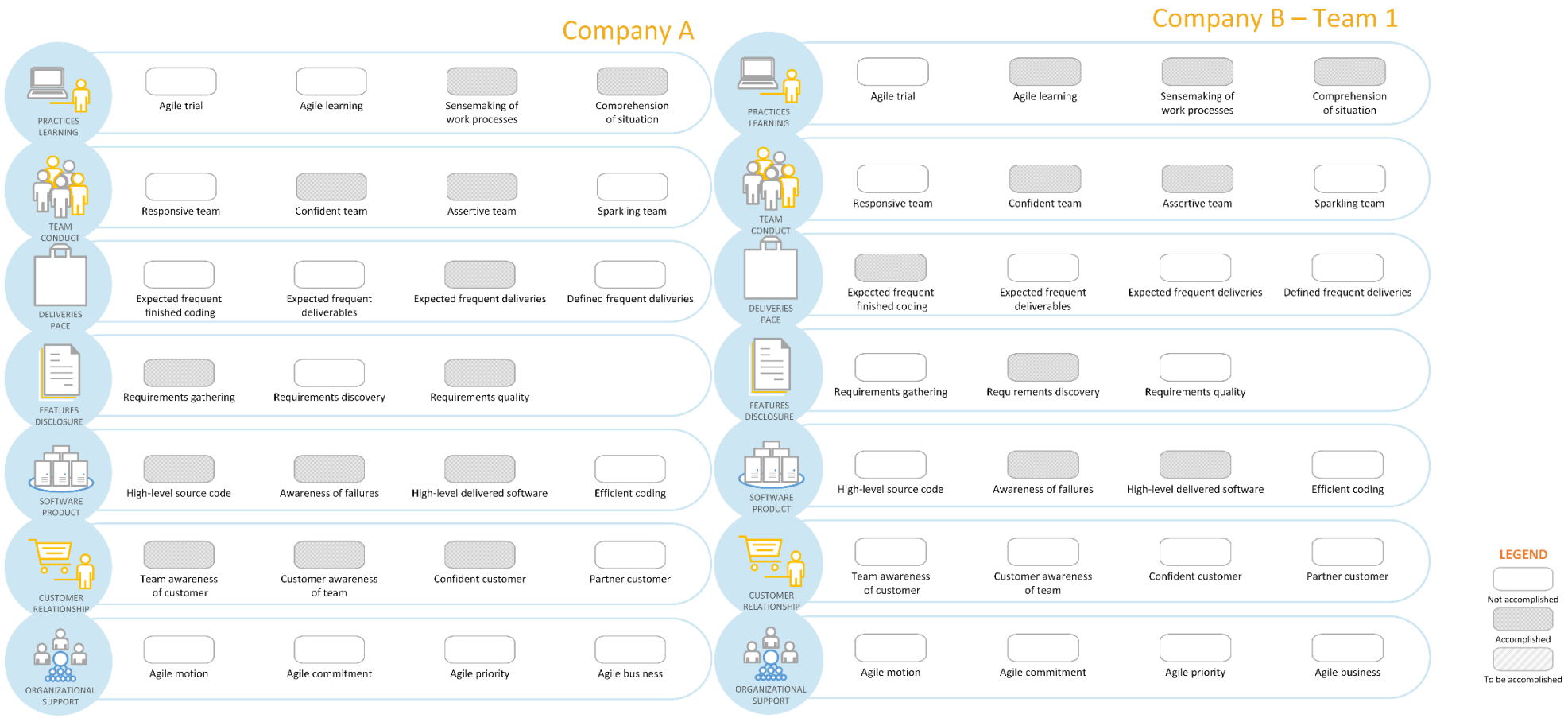
1. We identified that, in Company A's team, a Confident Team was an outcome in the PAST, based on the codes linked to this node on the map. At PRESENT, it evolved to an Assertive Team;
2. For Company B – Team 2, evidence from the PAST shows a Specialist and a Responsive Team (the codes that evidence them are respectively attached to the node). At PRESENT, the outcomes Inter-disciplinary Team, Confident and Assertive Team were identified.
3. For Company B – Team 1, evidence in the PAST led to the inference of a Confident Team, at PRESENT to an Assertive Team and, in the FUTURE, to an Specialist Team.
4. For Company C's team, note that in the past there is no evidence for outcomes on the team, but they appear at PRESENT as a Responsive Team and, in the FUTURE, as a Confident Team.

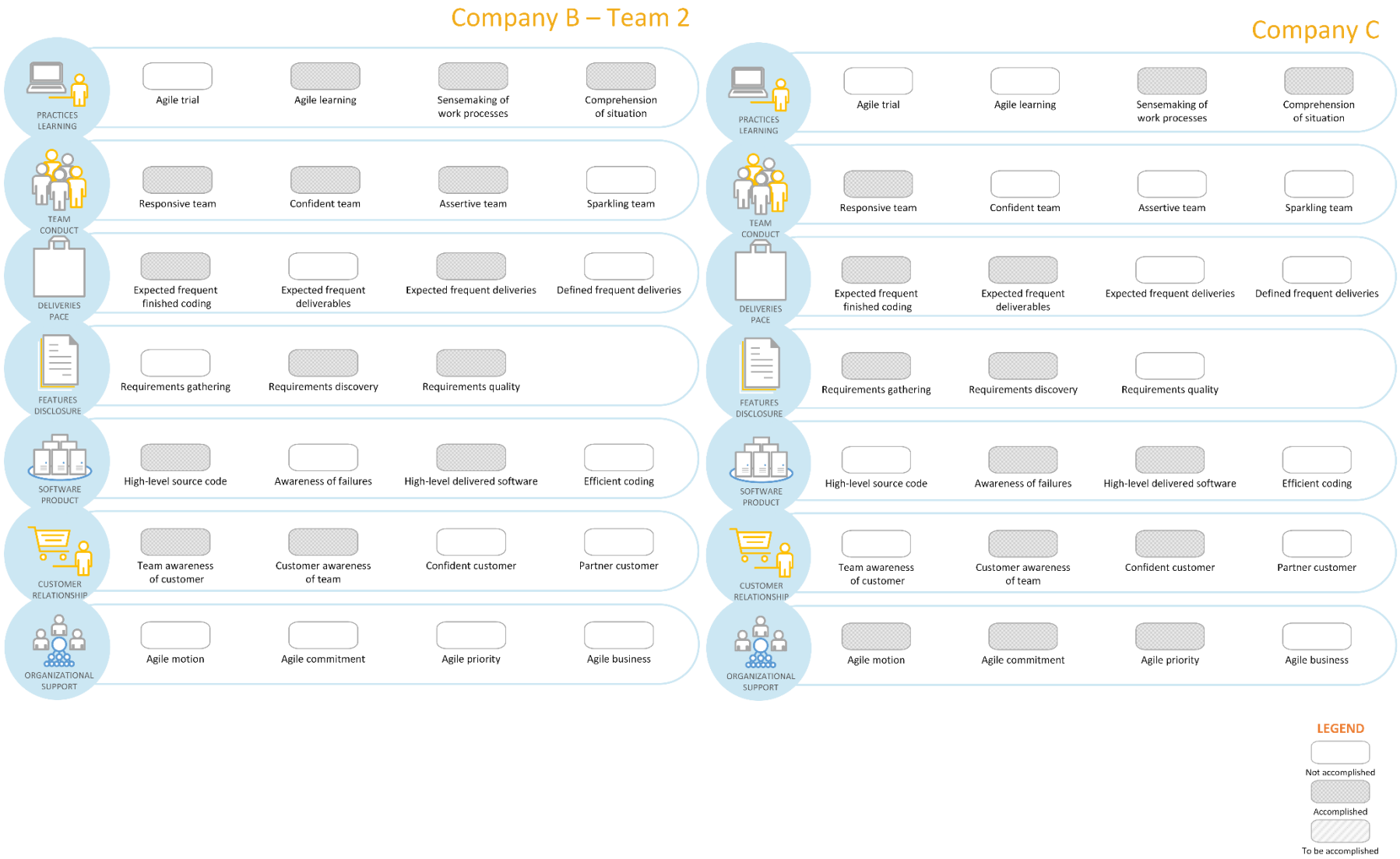
To build our cross-case map, we identified that, although a Responsive Team appeared at present for Company C, it is an outcome pursued before the Confident Team, which appeared at different moments in the cases. As Assertive Team was an outcome pursued after the Confident Team in the other cases, we inferred the sequence Responsive-Confident-Assertive Team. Note that the Specialist Team and Interdisciplinary team were not considered in the cross-case results, because of their context-specific evidence. This example also shows that the Team category of outcome emerged in the cross-case analysis; it was not pre-defined in the maps of individual teams.

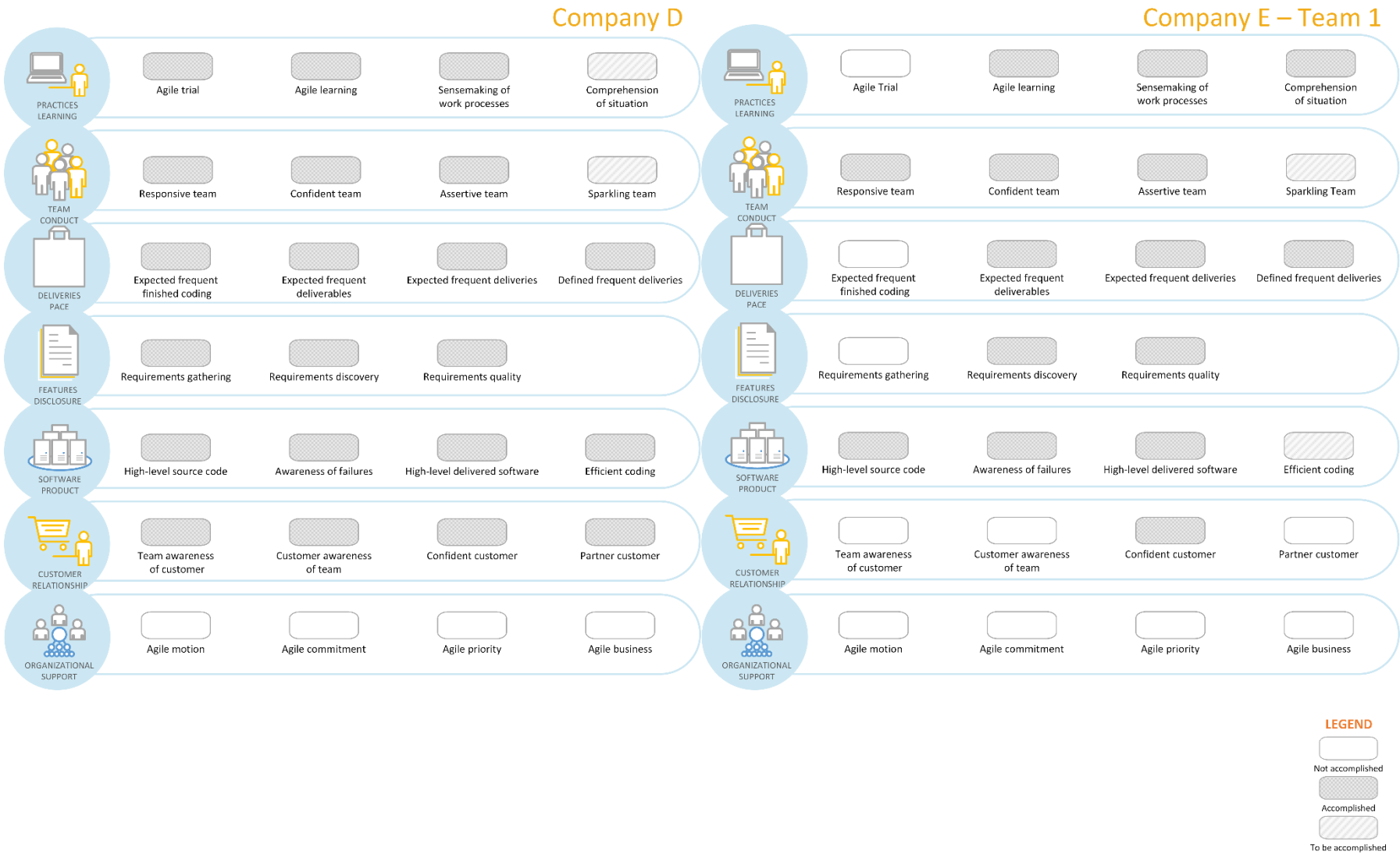
APPENDIX G – MULTIPLE CASES REPORT

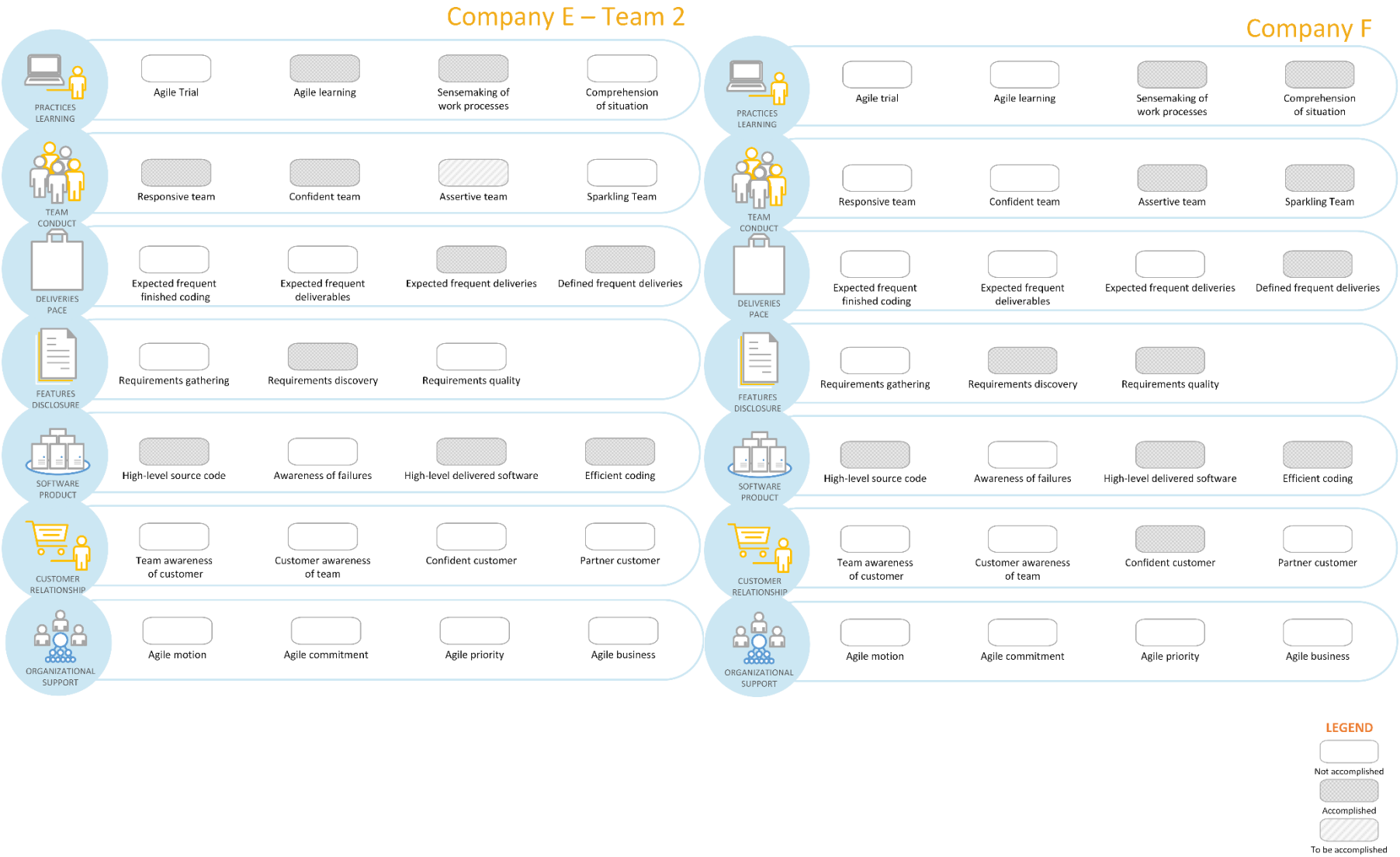


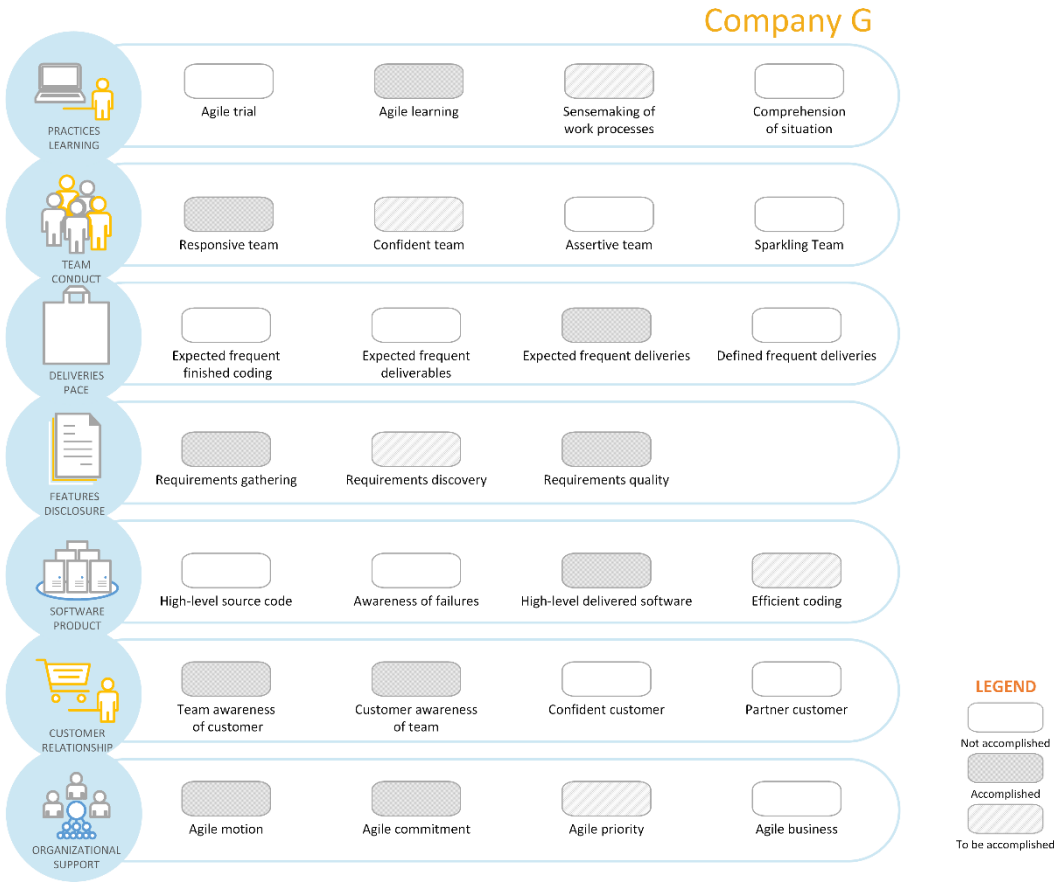
APPENDIX H – OUTCOMES ASSESSMENT FOR EACH TEAM




Company B – Team 2
PRACTICES LEARNING
Agile trial
Agile learning
Sensemaking of work processes
Comprehension of situation
TEAM CONDUCT
Responsive team
Confident team
Assertive team
Sparkling team
DELIVERIES PACE
Expected frequent finished coding
Expected frequent deliverables
Expected frequent deliveries
Defined frequent deliveries
FEATURES DISCLOSURE
Requirements gathering
Requirements discovery
Requirements quality
SOFTWARE PRODUCT
High-level source code
Awareness of failures
High-level delivered software
Efficient coding
CUSTOMER RELATIONSHIP
Team awareness of customer
Customer awareness of team
Confident customer
Partner customer
ORGANIZATIONAL SUPPORT
Agile motion
Agile commitment
Agile priority
Agile business
Company C
PRACTICES LEARNING
Agile trial
Agile learning
Sensemaking of work processes
Comprehension of situation
TEAM CONDUCT
Responsive team
Confident team
Assertive team
Sparkling team
DELIVERIES PACE
Expected frequent finished coding
Expected frequent deliverables
Expected frequent deliveries
Defined frequent deliveries
FEATURES DISCLOSURE
Requirements gathering
Requirements discovery
Requirements quality
SOFTWARE PRODUCT
High-level source code
Awareness of failures
High-level delivered software
Efficient coding
CUSTOMER RELATIONSHIP
Team awareness of customer
Customer awareness of team
Confident customer
Partner customer
ORGANIZATIONAL SUPPORT
Agile motion
Agile commitment
Agile priority
Agile business
LEGEND
Not accomplished
Accomplished
To be accomplished


Company D
PRACTICES LEARNING
Agile trial
Agile learning
Sensemaking of work processes
Comprehension of situation
TEAM CONDUCT
Responsive team
Confident team
Assertive team
Sparkling team
DELIVERIES PACE
Expected frequent finished coding
Expected frequent deliverables
Expected frequent deliveries
Defined frequent deliveries
FEATURES DISCLOSURE
Requirements gathering
Requirements discovery
Requirements quality
SOFTWARE PRODUCT
High-level source code
Awareness of failures
High-level delivered software
Efficient coding
CUSTOMER RELATIONSHIP
Team awareness of customer
Customer awareness of team
Confident customer
Partner customer
ORGANIZATIONAL SUPPORT
Agile motion
Agile commitment
Agile priority
Agile business
Company E – Team 1
PRACTICES LEARNING
Agile Trial
Agile learning
Sensemaking of work processes
Comprehension of situation
TEAM CONDUCT
Responsive team
Confident team
Assertive team
Sparkling Team
DELIVERIES PACE
Expected frequent finished coding
Expected frequent deliverables
Expected frequent deliveries
Defined frequent deliveries
FEATURES DISCLOSURE
Requirements gathering
Requirements discovery
Requirements quality
SOFTWARE PRODUCT
High-level source code
Awareness of failures
High-level delivered software
Efficient coding
CUSTOMER RELATIONSHIP
Team awareness of customer
Customer awareness of team
Confident customer
Partner customer
ORGANIZATIONAL SUPPORT
Agile motion
Agile commitment
Agile priority
Agile business
LEGEND
Not accomplished
Accomplished
To be accomplished


Company E – Team 2
PRACTICES LEARNING
Agile Trial
Agile learning
Sensemaking of work processes
Comprehension of situation
TEAM CONDUCT
Responsive team
Confident team
Assertive team
Sparkling Team
DELIVERIES PACE
Expected frequent finished coding
Expected frequent deliverables
Expected frequent deliveries
Defined frequent deliveries
FEATURES DISCLOSURE
Requirements gathering
Requirements discovery
Requirements quality
SOFTWARE PRODUCT
High-level source code
Awareness of failures
High-level delivered software
Efficient coding
CUSTOMER RELATIONSHIP
Team awareness of customer
Customer awareness of team
Confident customer
Partner customer
ORGANIZATIONAL SUPPORT
Agile motion
Agile commitment
Agile priority
Agile business
Company F
PRACTICES LEARNING
Agile trial
Agile learning
Sensemaking of work processes
Comprehension of situation
TEAM CONDUCT
Responsive team
Confident team
Assertive team
Sparkling Team
DELIVERIES PACE
Expected frequent finished coding
Expected frequent deliverables
Expected frequent deliveries
Defined frequent deliveries
FEATURES DISCLOSURE
Requirements gathering
Requirements discovery
Requirements quality
SOFTWARE PRODUCT
High-level source code
Awareness of failures
High-level delivered software
Efficient coding
CUSTOMER RELATIONSHIP
Team awareness of customer
Customer awareness of team
Confident customer
Partner customer
ORGANIZATIONAL SUPPORT
Agile motion
Agile commitment
Agile priority
Agile business
LEGEND
Not accomplished
Accomplished
To be accomplished


Company G
PRACTICES LEARNING
Agile trial
Agile learning
Sensemaking of work processes
Comprehension of situation
TEAM CONDUCT
Responsive team
Confident team
Assertive team
Sparkling Team
DELIVERIES PACE
Expected frequent finished coding
Expected frequent deliverables
Expected frequent deliveries
Defined frequent deliveries
FEATURES DISCLOSURE
Requirements gathering
Requirements discovery
Requirements quality
SOFTWARE PRODUCT
High-level source code
Awareness of failures
High-level delivered software
Efficient coding
CUSTOMER RELATIONSHIP
Team awareness of customer
Customer awareness of team
Confident customer
Partner customer
ORGANIZATIONAL SUPPORT
Agile motion
Agile commitment
Agile priority
Agile business
LEGEND
Not accomplished
Accomplished
To be accomplished

APPENDIX I – EXAMPLE OF EVIDENCES FOR THE AGILE COMPASS