The Journal of Systems and Software 102 (2015) 88–108

Contents lists available at ScienceDirect

The Journal of Systems and Software

journal homepage: www.elsevier.com/locate/jss

Progressive Outcomes: A framework for maturing in agile software development

Rafaela Mantovani Fontana [a,b,*], Victor Meyer Jr. [a], Sheila Reinehr [a], Andreia Malucelli [a]

[a] Pontifical Catholic University of Paraná (PUCPR), R. Imaculada Conceição, 1155, Prado Velho, 80215-901 Curitiba, PR, Brazil
[b] Federal University of Paraná (UFPR), R. Dr. Alcides Vieira Arcoverde, 1225, Jd. das Américas, 81520-260 Curitiba, PR, Brazil

ARTICLE INFO

Article history:
Received 25 August 2014
Revised 2 December 2014
Accepted 15 December 2014
Available online 22 December 2014

Keywords:
Maturity
Agile software development
Software process improvement
Complex adaptive systems
Ambidexterity

ABSTRACT

Maturity models are used to guide improvements in the software engineering field and a number of maturity models for agile methods have been proposed in the last years. These models differ in their underlying structure prescribing different possible paths to maturity in agile software development, neglecting the fact that agile teams struggle to follow prescribed processes and practices. Our objective, therefore, was to empirically investigate how agile teams evolve to maturity, as a means to conceive a theory for agile software development evolvement that considers agile teams nature. The complex adaptive systems theory was used as a lens for analysis and four case studies were conducted to collect qualitative and quantitative data. As a result, we propose the Progressive Outcomes framework to describe the agile software development maturing process. It is a framework in which people have the central role, ambidexterity is a key ability to maturity, and improvement is guided by outcomes agile teams pursue, instead of prescribed practices.

© 2014 Elsevier Inc. All rights reserved.

1. Introduction

Maturity models are tools to describe how an element evolves. This element, which may be a person, an object, or a social system, may use the description provided by maturity models to assess its own situation and find guidance to improve in a specific focus area (Kohlegger et al., 2009). In the software engineering field, process improvement is based mainly on the guidelines given by the Capability Maturity Model Integration for Development –CMMI-DEV (CMMI Product Team, 2010) and the international standard ISO/IEC 15504 (ISO/IEC, 2004). Both CMMI-DEV and ISO/IEC 15504 share the underlying assumption that organizational capabilities must be codified in processes that are previously planned and designed (Maier et al., 2012). The improvement for the organization comes from the definition, institutionalization and quantitative management of these processes (CMMI Product Team, 2010).

A number of agile software development teams have been implementing these process improvement initiatives (Al-Tarawneh et al., 2011; Anderson, 2005; Baker, 2006; Caffery et al., 2008; Cohan and Glazer, 2009; Jakobsen and Johnson, 2008; Lina and Dan, 2012; Lukasiewicz and Miler, 2012; Spoelstra et al., 2011; Sutherland et al., 2007; Tuan and Thang, 2013). The benefits of such initiatives have been recognized as a "magic potion", as they provide a powerful combination of adaptability and predictability (Sutherland et al., 2007). However, if teams are meant to keep agile in the highest maturity levels, the improvement path cannot be based on current established maturity models. The increasing processes definition hinders sustaining agility (Lukasiewicz and Miler, 2012; Paulk, 2001).

If agile methods place people and interaction over processes and tools (Beck et al., 2001; Conboy et al., 2011), the improvement road map for these methods should not be based on processes definition (Fontana et al., 2014). There are, for this reason, a number of agile maturity models proposed in the literature (Leppänen, 2013; Ozcan-Top and Demirörs, 2013; Schweigert et al., 2012). They are built over agile values and the improvement paths they suggest consider sustaining agility in the highest maturity levels. Two issues linger, though: the first is that they prescribe the practices the team should implement, even agile teams, which consider their work as a highly context-specific job to be prescribed (Fontana et al., 2014; Kettunen, 2012; Schweigert et al., 2012; Sidky et al., 2007); and the second is that the models still differ in their proposals, which indicates that the path to maturing in agile software development has not been uncovered yet.

These two issues thus, motivated this study. Our objective was to identify how agile software development teams evolve to maturity. We conducted an empirical research, through a multiple-case study approach, that identified how real agile teams evolve their practices and mature over time. The findings interest researchers, as they innovate in the underlying theory for a maturity model; and practitioners, as they provide practical guidelines for improving agile methods.

* Corresponding author at: Federal University of Paraná (UFPR), R. Dr. Alcides Vieira Arcoverde, 1225, Jd. das Américas, 81520-260 Curitiba, PR, Brazil. Tel.: +55 4133614904.
E-mail addresses: rafaela.fontana@ufpr.br (R.M. Fontana), victormeyerjr@gmail.com (V. Meyer), sheila.reinehr@pucpr.br (S. Reinehr), malu@ppgia.pucpr.br (A. Malucelli).

http://dx.doi.org/10.1016/j.jss.2014.12.032
0164-1212/© 2014 Elsevier Inc. All rights reserved.

This paper is organized as follows: Section 2 outlines related work; Section 3 discusses our theoretical foundation and the theoretical framework we used for data analysis; and Section 4 presents the research structure. The results of the multiple-case study are presented in Section 5 and, finally, the findings are discussed and concluded in Sections 6 and 7, respectively.

2. Related work

Improvement paths in software engineering are currently defined by the guidelines given by CMMI-DEV and the international standard ISO/IEC 15504. The standard defines that software process improvement is accomplished through implementation of processes that address software acquisition, supply, engineering, operation, management, improvement, resources and infrastructure, reuse, configuration control, quality assurance and product quality (ISO/IEC, 2004).

Besides the incremental implementation of such processes, the capability to execute the processes should follow an evolutionary path. In ISO/IEC 15504 this capability is defined in terms of six levels: incomplete process, performed process, managed process, established process, predictable process and, lastly, optimizing process (ISO/IEC, 2004).

CMMI-DEV defines similar capability levels and, in addition, describes maturity levels to be followed for process improvement (CMMI Product Team, 2010). These levels are defined in terms of a set of processes that should be implemented and by the capability in which these processes should be performed. The first maturity level is named Initial, as processes here are usually ad hoc and chaotic; the second is called Managed, characterized by processes that lead projects to be performed and managed according to their documented plans. Next, comes third stage, Defined, in which processes are well characterized, understood and standardized in the organization. In fourth stage, "the organization and projects establish quantitative objectives for quality and process performance and use them as criteria in managing projects" (CMMI Product Team, 2010, p. 28) and for this reason it is called Quantitatively Managed. The highest maturity level is the Optimizing, in which improvement of the processes is continuous and based on a quantitative understanding of objectives and needs.

In the same path of CMMI two other models were created by Watts Humphrey: Personal Software Process (PSP) and Team Software Process (TSP). The first focuses on software individual discipline and the latter focuses on a disciplined group formed by PSP practitioners (Humphrey, 1995). The main difficulty that one faces when trying to use PSP is the amount of self-discipline that is required to fully apply the method. The first challenge is the need to record every single activity that the software engineer performs, the time that was spent in the activity and all the breaks taken during the work. This approach creates a bureaucracy in daily work that does not match agile approaches emphasis on people and interaction. PSP is also based on a seven-step roadmap that leads developer from a complete immature state (PSP0 – Baseline Personal Process) to a full mature state (PSP3 – Cyclic Personal Process). Similarly, TSP is a framework to be used in teams composed by PSP trained members (Humphrey, 2010). The model is based on an eight-step cycle: launch, strategy, plan, requirements, design, implementation, test and postmortem. The difficulty with TSP is the same of the PSP: the bureaucratic and prescriptive way to evolve work processes.

In the field of agile software development, two main lines of research have been studying maturity. The first focuses on adapting agile practices and principles to fit current software maturity models, such as CMMI-DEV. The second focuses on creating maturity paths related to agile software development values.

Since researchers and practitioners consider agile methods and software process improvement models as means to get the best from software development, there has been an increasing interest in combining both approaches. Starting with Mark Paulk explaining how Extreme Programming could be complementary to CMM (Paulk, 2001), a number of studies have either reported how companies have combined agile methods to CMMI-DEV requirements, or proposed new approaches to perform this combination (Al-Tarawneh et al., 2011; Anderson, 2005; Baker, 2006; Caffery et al., 2008; Cohan and Glazer, 2009; Jakobsen and Johnson, 2008; Lina and Dan, 2012; Lukasiewicz and Miler, 2012; Sutherland et al., 2007; Spoelstra et al., 2011; Tuan and Thang, 2013).

All these approaches result in adapting agile methods to fit assessment requirements in CMMI-DEV. They also accept that agile methods do not fit higher maturity levels, as the quantitative control of processes does not apply to agility (Lukasiewicz and Miler, 2012; Paulk, 2001). However, these initiatives recognize the value of having a combination of agility and disciplined processes (Boehm and Turner, 2004).

The second group of studies considers maturity in agile software development by keeping its focus on agility. The first agile maturity model we found published in the literature was the one by Nawrocki et al. (2001). They present a 4-level maturity model for XP (Extreme Programming). The authors' motivation was to identify if a specific organization is using XP or not. Typically, there are assessment problems because XP practices are not fully applied all the time. Thus, they say that a maturity model could be used as a reference point. For them, the model has to be hierarchical and identify practices for each level, such as CMMI-DEV.

The proposed model is called XPMM (eXtreme Programming Maturity Model) and was created based on intersections between XP and CMMI-DEV. The first level, named "Not compliant at all" means that the team applies no or a few XP practices. The second, "Initial", focuses on project teams and defines two process areas: Customer Relationship Management and Product Quality Assurance. The third, "Advanced" is focused on coding and, thus, the only process area is Pair Programming. The last level, called "Mature", has process areas related to the satisfaction of customers and developers. Here, the only process area is Project Performance. To be assigned to a specific level, the team has to follow the practices in this level and in the previous ones.

The authors assume that assessment should not be based on rich process documentation. They also state that many XP practices (e.g. simplicity, no functionality is added early) are difficult to assess. Then, they propose conversation and observation as assessment methods.

Another proposal for XP maturity model is the one presented by Lui and Chan (2005). They report a roadmap to aid the adoption of XP in Chinese inexperienced teams. They have analyzed the dependencies among XP practices and mapped them into a matrix. With the help of a visual data mining tool, they identified the ideal sequence for practices implementation.

They arrived at a four-stage XP implementation road map. In stage 1, the team should implement testing, simple design, refactoring and coding standard. In stage 2, the team should focus on continuous integration. In stage 3, the team should implement pair programming and collective ownership and, finally, in stage 4, the remaining XP practices would be adopted: metaphor, 40-h week, small release, on-site customer and planning game.

The model presented by Packlick (2007) is based on a single experience in Sabre Airline Solutions. He describes a different approach, based on the observation of real teams. The proposal is to use a goal-oriented approach because the author has observed that teams got more motivated to find out their own ways to get the job done. The AGILE Roadmap comprises five maturity levels that represent different learning stages an agile team should accomplish.

The goals are related to the AGILE acronym: Acceptance criteria; Green-bar tests and builds; Iterative planning; Learning and adapting; and, Engineering excellence. Each goal is detailed as a user's story

that describes what teams must accomplish and the agile roadmap is defined by the intersection of such goals and maturity levels. The first maturity level is "Awareness", when the team understands the goals and the value to accomplish them. The second, "Transformation", is when knowledge is put into practice on a regular basis. The third level is "Breakthrough", when practices are kept even under pressure. The next, "Optimizing" level, in which improvements are continuous (creative innovation) and, the last, "Mentoring" is when coaching is performed by high performance teams to share knowledge in the company.

The model proposed by Sidky et al. (2007) is part of a whole framework to guide and to assist agile adoption. The framework comprises an agility measurement index and a four-stage process to introduce the practices in the organization. The measurement index (Sidky Agile Measurement Index – SAMI) defines five agile levels named 1) Collaborative, 2) Evolutionary, 3) Effective, 4) Adaptive and 5) Encompassing. Each level is defined in terms of agile practices that should be adopted according to agile principles. The first level focuses on enhancements in communication and collaboration; the second focuses on software delivery – early and continuously; the third, focuses on software quality; the fourth level has practices to develop the capability of responding to changes; and the last focuses on creating an environment to sustain agility.

Qumer and Henderson-Sellers (2008) also proposed a framework for agile adoption and the Agile Adoption and Improvement Model (AAIM) is part of this framework. It defines six agile stages grouped in three blocks. The first block – Prompt – has level 1) Agile Infancy. The second block – Crux – comprises levels 2) Agile Initial, 3) Agile Realization and 4) Agile Value. The last block – Apex – has the two last levels: 5) Agile Smart and 6) Agile Progress.

AAIM Level 1 is focused on introducing basic agile properties (speed, flexibility, and responsiveness). At AAIM Level 2, the focus is to enable communication and collaboration. The next, AAIM Level 3 is represented by the use of executable artifacts and minimal documentation. At AAIM Level 4, the agile basis is established and the focus is on valuing people and not ignoring tools and processes. AAIM Level 5 focuses on learning and, lastly, AAIM Level 6 establishes a lean environment, with high quality and minimum resources, sustaining agility.

Patel and Ramachandran (2009) presented a proposal based on the CMMI-DEV structure, but process areas and practices focus on agile principles. Their levels are Initial, Explored, Defined, Improved and Sustained. Maturity gain, as in CMMI-DEV, is related to increasing process definition and control through metrics, based on agile practices.

At the Initial level, the development environment is unstable. The next level, Explored, means implementing project planning, story cards-driven development, on-site customer and introduction to test-driven development. The third level, named Defined, focuses on customer satisfaction, communication enhancements, software quality and improving coding practices. At the fourth level, Improved, the objectives are: measuring the software process, achieving empowered teams and rewards, implementing project management, assessing project risks and working with simplicity and no working overtime. The highest maturity team, at the Sustained level, continuously improves software process, managing uncertainties, tuning project performance and preventing defects on software.

Benefield (2010) presents his experience at British Telecom in defining seven practices – or dimensions – that, together, lead to significant improvement in agile solutions: automated regression testing; code quality metrics; automated deployment and backout; automated builds and configuration management best practices; interlocked delivery and interface integration testing; test driven development; performance and scalability testing.

These dimensions are implemented through five levels that define to what extent the practices are adopted within the team, or across teams. Level 1 is called Emergent Engineering Best Practices, Level 2 is called Continuous Practices at Component Level, Level 3 represents Cross Component Continuous Integration, Level 4 is called Cross Journey Continuous Integration and Level 5 is called On Demand Just in Time Releases. At this higher level, teams are highly productive and new products are quickly delivered through a service-oriented architecture.

The only model based on Scrum adoption is the one presented by Yin et al. (2011): the Scrum Maturity Model. The purpose of the model is to aid organizations to improve software process based on the customer and to help the adoption of Scrum in a staged and incremental manner. It comprises five levels defined with goals, objectives, specific and suggested practices.

Level 1, Initial, represents the absence of goals for process improvement. At Level 2, the Managed, Scrum practices are more structured and complete. At Level 3, Defined, the focus is on the relationship with customers. At Level 4, Quantitatively Managed, there are metrics and management of the process performance. Lastly, at Level 5, Optimizing, the focus is on performance management.

Our previous work (Fontana et al., 2014b) also presented a sequence for the adoption of agile practices, according to practitioners' perception. The agile software development maturing process would start with agile values, involved customer, agile planning and agile requirements. Then, in a second stage, the focus would be more technical, with the implementation of agile coding and agile testing practices. In this previous work, we also identified that a number of practices could be implemented at any time (such as software architecture, agile physical environment, agile quality assurance and agile project monitoring). The metrics, the definition of processes and the control of processes were found to be optional practices to mature in agility.

These nine maturity models for agile software development present quite different structures, which suggest that maturity in agile software development is still being defined. Various proposals are still only published on the Internet (Buglione, 2011; Schweigert et al., 2012). Scientifically tested models are either based on experiences of specific cases, specific methods, or focus on the adoption of agile methods.

To provide summarized evidence, we analyzed each of these nine models according to the directions presented by Maier et al. (2012, p. 149) for planning and developing of maturity models. The result 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 specific to a domain;
- 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: process areas uncover the conceptual foundation used in the development of the model and must be "mutually exclusive and collectively exhaustive" (Maier et al., 2012, p. 150);
- Maturity levels: they 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 in the intersection of process areas and maturity levels;
- Administration mechanism: definition of the mechanisms to apply 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

Table 1
Analysis of the structure of agile maturity models.

	Audience	Aim	Scope	Success criteria	Process areas	Maturity levels	Cell texts	Administration mechanism
Nawrocki et al. (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 and 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 et al. (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 and 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 and 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 et al. (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 not shown on paper	Partial. Briefly describes six appraisals
Fontana et al. (2014b)	Agile teams	Analytic	Agile in general	No evidence	No, argues that agile teams dislike prescription of practices	3 levels, with processes that are optional and others to be implemented anytime	No evidence	No evidence

on existence and adherence to a structured process, probably because of the influence of CMMI-DEV.

Although agile methods focus on people interaction over processes and tools (Beck et al., 2001) and software process improvement guidelines for agile methods should be built over people, and not processes (Fontana et al., 2014), Table 1 shows that most of current agile maturity models still focus on adherence to structured process. The only author that explicitly describes how team evolves is Packlick (2007). He also considers the need to allow for self-organization in the team, which is an agile principle (Beck et al., 2001).

His approach to describe how an agile team evolves diverges from values that underlie TSP and PSP guidelines. These guides were developed to create an environment that enables software practitioners and teams to evolve to a maturity state in which they plan and track their work using a commonly accepted set of measurements and standards (Humphrey et al., 2010; Pomeroy-Huff et al., 2005). Using guidelines given by PSP, individuals increasingly measure, plan and improve their own work based on quantitative data (Pomeroy-Huff et al., 2005). TSP argues that in a software team, members should follow PSP and, accordingly, define and use structured processes, collect data, analyze data and use statistical foundation for process improvement in the team (Humphrey et al., 2010). In comparison with the highest maturity level proposed by Packlick (2007), in which mature teams focus on creative innovation and mentoring, there is clearly a mismatch between the approaches.

According to TSP, the maturing process of the team starts in a forming stage, in which the team comes together as a working group and team members are highly dependent on leader. This team goes then to a storming stage in which members start to claim for the position among them and conflicts may occur. If 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 to performing tasks. Although TSP argues that software teams should self-manage (Humphrey et al., 2010, p. 14), it describes an evolvement path based on Tuckman's model for team development (Tuckman, 1965), which has currently been recognized as hardly applicable to self-managing work teams (Kuipers and Stoker, 2009).

Instead of following a linear path of development, Kuipers and Stoker (2009) show that self-managing work teams develop through three team processes that occur simultaneously: task management, in which they develop activities linked to "team multifunctionality and its capabilities to manage responsibilities and control" (Kuipers and Stoker, 2009, p. 407); internal relations, in which team deals with internal cooperative issues; and external relations and improvement, in which team deals with relationship with other teams, customers and suppliers. These teams evolve to greater self-management and, thereby, increased performance and enhanced quality of working life.

This last approach seems to be more suitable to agile software development teams, since agility encourages self-organizing and self-managing behavior (Hoda et al., 2012; Moe et al., 2009), instead of stimulating personal and team process definition and control. This emphasis on people, interactions and self-organization lead to the consideration that agile software development teams present a number of emergent capabilities, which characterize them as complex adaptive systems (Augustine et al., 2005; Hoda et al., 2012; Vidgen and Wang, 2009). It brings to light some considerations about the management tools that are applicable to agile teams and, we argue, for process improvement guidelines for agile teams, as shown in the next section.

3. Theoretical foundation

Agile software development teams are complex adaptive systems (Power, 2014; Vidgen and Wang, 2009). They consist of a number of connected agents that "interact with each other according to sets of rules that require them to examine and to 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). These interactions are not visible and not immediately understood (Perrow, 1981) and complex behavior emerges from interrelationships, interaction and interactivity of the elements or among the elements and the environment (Mitleton-Kelly, 2003).

These interactions create the "shadow networks" of the system, where links between individuals are spontaneously and informally established (Stacey, 1996). These networks define most of the decisions made by the system and, for this reason, organizations are not defined by those "charts [. . .] printed down on an organization meeting room", but by the ones that emerge from informal interactions (Hidalgo, 2011, p. 557). It leads to the acknowledgment that practices in these systems are not codified, but emerge from praxis, as people hardly-ever accomplish their jobs exactly as designed by the system (Campbell-Hunt, 2007; Stacey et al., 2000).

It challenges management, thus, to give to some in the organization a position responsible for shaping this emergent order (Campbell-Hunt, 2007). Complex adaptive systems cannot be managed as "machine-like" systems, as it leads to trials to make workers accomplish goals defined by management, and with no deviation from plans (McDaniel Jr., 2007). Instead, complex systems must be managed with processes of sensemaking (Weick et al., 2005), learning from experience, and improvisation to deal with uncertainty (McDaniel Jr., 2007). In complex contexts, things are understood only in retrospect and patterns may then emerge if leaders conduct experiments that are safe to fail. Decision making may thus be based on probing, sensing and responding (Snowden and Boone, 2007).

Our assumption is that the tools management should use for software process improvement should also shift. Maturity models such as CMMI-DEV, ISO/IEC 15504, PSP and TSP focus on codifying work processes to reduce uncertainty. However, in complex systems uncertainty must be accepted (Tsoukas, 2005). As we have just recognized agile teams as complex adaptive systems, we need to understand how these systems behave so that we can build improvement guides that consider this complex nature.

Complex adaptive system theory shows that complex systems evolve in a discontinuous way and this process involves instability (Eijnatten, 2003). This instability must be combined with stability to generate a space called "edge-of-chaos" (Stacey, 1996, p. 97), in which creativity and novel solutions emerge. Studies show, for example, that dynamic capabilities in high-velocity markets are developed in an "unstable state of slipping into too much or too little structure" (Eisenhardt and Martin, 2000, p. 1113).

Accordingly, March (1991) identified that for an adaptive system to survive and to prosper, it needs to balance processes of exploration and exploitation. Exploration includes things captured by terms such as "search, variation, risk taking, experimentation, play, flexibility, discovery, innovation". On the other hand, exploitation includes such things as "refinement, choice, production, efficiency, selection, implementation, execution" (March, 1991, p. 71). The successful combination of these processes has been recognized as organizational ambidexterity. Higher levels of ambidexterity have already been related to higher levels of performance (Gibson and Birkinshaw, 2004) and to the ability of the organization to adapt over time (O'Reilly III and Tushman, 2008).

Ambidextrous organizations are aligned and efficient in the management of current demands while simultaneously adaptive to change in the environment (Raisch and Birkinshaw, 2008). They have "the ability to both use and refine existing knowledge (exploitation) while also creating new knowledge to overcome knowledge deficiencies or absences identified within the execution of the work (exploration)" (Turner et al., 2013, p. 320).

Ambidexterity in organizations have been identified, for example, by analyzing the perception of alignment – whether people in the

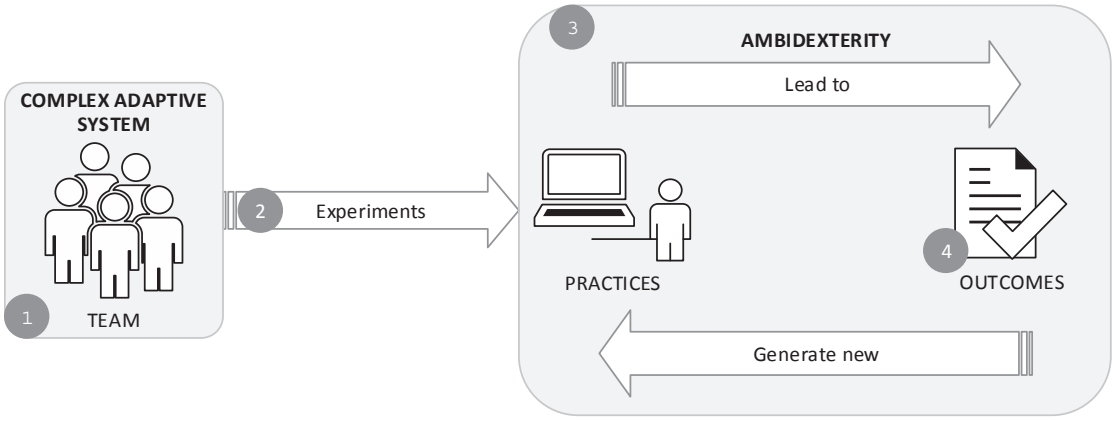

Fig. 1. Theoretical framework.

organization work towards the same goals – and adaptability – the capacity to reconfigure activities to meet changing demands (Gibson and Birkinshaw, 2004). Information flows among individuals, which Tiwana (2008) calls "ties", also characterize ambidexterity. It is accomplished in a successful combination of strong ties – sharing of common language, values and cooperative norms and bridging ties – heterogeneity in experiences, knowledge and skills (Tiwana, 2008).

The essence of this theoretical foundation is, thus, that agile software development teams are complex adaptive systems. They are driven by a self-organized behavior, which challenges the management to use 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 a higher performance through ambidextrous abilities. Next, four statements summarize our theoretical basis for data analysis, as shown in Fig. 1.

1) An agile software development team is a complex adaptive system that evolves in a discontinuous way;
2) Improvement in the work processes is performed through experimentation, which is a way to probe new solutions;
3) High performance is achieved through ambidexterity; and
4) To accomplish that, teams develop dynamic capabilities pursuing outcomes, and not following codified routines.

4. Research approach

The objective of this study was to identify the mechanisms teams apply to mature in agile software development. Our research question was thus "how do teams mature in agile software development?" To answer this question, we chose a multiple-case study approach. Case study is a research strategy to understand the dynamics present within single settings (Eisenhardt, 1989) and appropriate to answer the "how" questions in research (Yin, 2005).

The unit of analysis was the agile software development team. There were no specific choice criteria for teams: they could have any size, apply any method, no matter how long agile had been adopted. We were searching for a pattern of evolution in agile teams and, the most diverse these teams were in context, the better. The lack of preference for a specific agile method, such as Scrum or XP, is due to the fact that agile methods are currently being highly tailored (Bustard et al., 2013) and current and future research in agile methods are expected not to focus on a single method (Kurapati et al., 2012).

In a case study, a researcher may define a priori the constructs to shape the research and to have a firmer empirical grounding (Eisenhardt, 1989). Thus, we defined our constructs – or propositions (Yin, 2005) – based on the theoretical framework presented in Fig. 1 and on our previous studies. We defined four propositions to be analyzed in the cases presented in Fig. 2. The figure shows the proposition, the theoretical foundation point it relates to and the authors that provide a theoretical basis to it.

The study included four agile teams in Brazil with the profiles presented in Table 2. We collected qualitative data – through interviews – and quantitative data – through questionnaires. Table 2 also shows the number of interviews and the period of data collection from each team.

We chose companies looking for different profiles of software development (size, type of customer, company main activities, project duration), to search for evidences of replication in different contexts (Eisenhardt, 1989). The primary activity of a company in the software development industry in Brazil is characterized by either 1) the development of software under demand; or 2) the development of customizable software; or 3) the development of non-customizable software; or 4) web portals and internet information services (Softex, 2012). In our research, we included teams that match the first three categories of software development.

The overall process of our research followed the stages suggested by Eisenhardt (1989). We started with data collection, within-case analysis and case report for each of the four teams. After all cases were reported individually, we cross-analyzed the results and generated the multiple-case report. All the team leaders we interviewed validated the report and, finally, we consolidated the results. Fig. 3 presents the overall research approach and the next subsections explain the details of each stage.

4.1. Data collection

Data collection was performed through interviews with three team members: the team leader, the most experienced developer and the least experienced developer. One exception was Team 2 in Company B, in which the least experienced developer was not available. The interviews took about 1 h each. We asked respondents to tell us a story of how their agile practices evolved over time. For each team, the three interviews complemented each other. We used the framework proposed by Kirk and Tempero (2012) as a guide for data collection. Then, the practices described in the interviews were assigned for 1) defining, 2) making or 3) delivering the software. As the interviewee told us a story (Tsoukas and Hatch, 2005), we also placed the practices they described in the past, in the present or in the future.

As a complementary quantitative approach for data collection, all the team members answered a questionnaire. Two types of questionnaire were applied: the first was answered by the leaders and comprised four parts: personal information, company information, ambidexterity information and project success perception. The second type of questionnaire was answered by the other team members and comprised two parts: personal information and ambidexterity information.

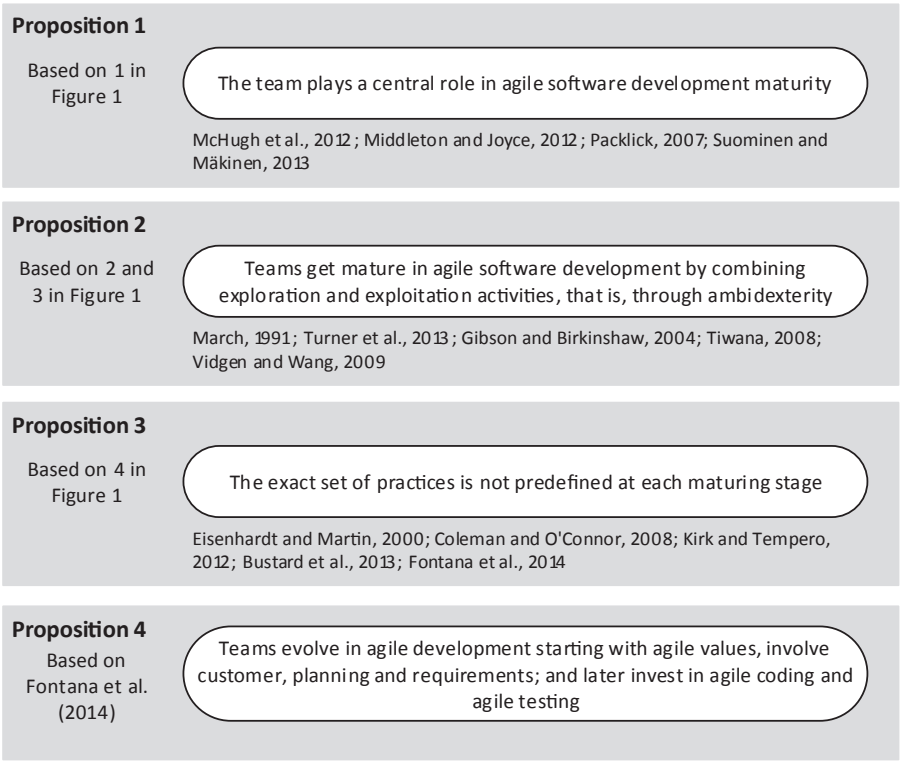

Fig. 2. Propositions of the study.

Table 2
The companies profile.

Team Alias	Profile		Number of interviews	Period
Company A	Team size	8 people	3	March, 2014
	Company size	100 people		
	Main activity	Document 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	3	April, 2014
	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	Under demand		
Company B – Team 2	Team size	7 people	2	April, 2014
	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	3	April, 2014
	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		

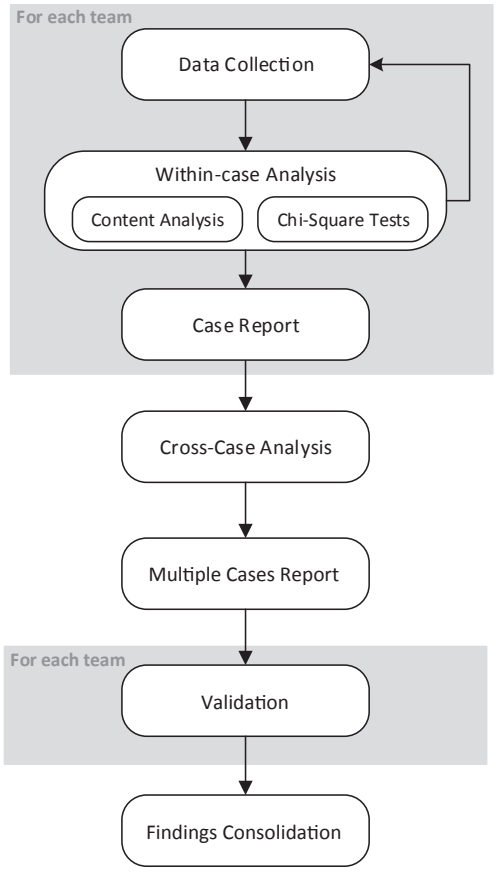

Fig. 3. Research approach.

All the questions regarding ambidexterity information and project success perception were answered in a five-point Likert scale (from 1 – completely disagree to 5 – completely agree). The ambidexterity questionnaire evaluated performance, alignment and adaptability (according to the study performed by Gibson and Birkinshaw, 2004) and bridging and strong ties (according to the study performed by Tiwana, 2008). The statements of the questionnaire are presented in Table A.1 in Appendix A. As suggested by Leppänen (2013), maturity studies should include the project success perception. Thus, we included this evaluation of the perceived projects success, based on the agile methods success factors identified by Misra et al. (2009).

4.2. Within-case analysis

Our within-case analysis included a qualitative and a quantitative data analysis. For the qualitative data analysis, all the interviews were recorded and transcribed. The documents were stored in Atlas TI, which was the tool used to apply the content analysis technique (Bardin, 2011).

According to Bardin's (2011) guidelines, for each interview, we performed the following steps: 1) scanning text; 2) codifying; 3) assigning each code to a category: define, make or deliver (Kirk and Tempero, 2012); 4) assigning each code to either past, present or future; 5) creating memos when necessary and 6) association with other related codes.

While categorizing the codes for a team as a "define", "make" or "deliver" practice (step 3), we also created three networks of codes for each team (step 4): one network for practices in the past, one for practices in the present and one for practices in the future (the network for Company A – Past is shown in Fig. B.1 in Appendix B). After all the practices were mapped, we identified the outcomes the team was pursuing with specific sets of practices. Thus, we mapped all these pursued outcomes and kept a track of which practices led us to the inference of each outcome. All this within-case information was recorded in the Case Report (see Fig. 3).

For the quantitative data analysis, the questionnaires were transcribed to spreadsheets and the data was consolidated. For each aspect evaluated of ambidexterity (performance, alignment, adaptability, bridging ties and strong ties), the percentage of responses given in the five-point Likert scale was grouped as "Disagree", when the classification was 1 or 2, "No opinion", when the classification was 3 and "Agree", for classifications 4 or 5.

Our objective with this analysis was to identify the ambidexterity perception in the team. Thus, we performed statistical Chi-Square tests for each frequency distribution considering the following null hypothesis:

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

All the null hypothesis for which the tests resulted in p-value smaller than 0.05 were rejected. In the Case Report, to present to the team leaders, we created graphs for each aspect evaluated, as shown in Fig. C.1 in Appendix C, the example of Company C.

In the quantitative approach, we pursued evidences of ambidexterity abilities in the team. To accomplish that, we expected all of the null hypothesis to be refused. That is, the whole team should be in accordance with the perception of performance, alignment, flexibility, strong ties and bridging ties. In addition to that, the highest percentage of respondents should agree positively with each of the evaluated aspects.

4.3. Cross-case analysis

Eisenhardt (1989) suggests that, in the cross-case analysis, the researcher performs the iterative tabulation of evidence for each construct and searches for replication across cases. We conducted this analysis with the help of a mind mapping tool. Fig. D.1 in Appendix D shows part of the mind map we used to record the outcomes across cases.

The outcomes identified in each within-case analysis were inserted in the map and the evidences to the original team were duly 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 categories of outcomes emerged: practices, team, deliveries, requirements, product and customer. For each category of outcome and their related sequence of outcomes, we traced the ones that each team had accomplished and included an overall view in the Case Report for the validation of the team leader.

4.4. 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 little changes. The transcription of the validation interviews was also stored in Atlas TI, codified and, whenever necessary, code networks and individual cases reports were updated.

4.5. Threats to validity

The quality of a case study research may be evaluated under four criteria: construct validity, internal validity, external validity and reliability (Yin, 2005). One threat to the construct validity in our study was the use of narratives to identify the agile practices evolvement. We recognize that the narrative approach for research is compatible with the need to appreciate the complexity of organizations (Tsoukas and 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.

Another limitation of the narrative approach is that people usually rationalize facts while they tell the story. It is the sensemaking of the facts that makes individuals interpret past facts and try to find explanations for what happened (Weick et al., 2005) and maybe blur what really happened. To lessen this threat, during the interviews, we asked practitioners to show us documents and tools they said they were using as a means to confirm the interpretation they gave for the facts. Likewise, the validation we performed with the team leaders also helped validating our perception of their interpretations of the sequence of facts.

Internal validity evaluation is applicable in our study regarding the inferences we performed to identify the outcomes the teams pursue. There is a threat to the validity when one considers the subjectivity involved in the interpretation of the practices. Our initiative to lessen this threat was conducting the validation of the research report with the team leaders.

With respect to external validity, our approach was to pursue analytical generalization (a trial to generalize a particular set of results to a broader theory), using replication logic in the multiple-case study (Yin, 2005). We grounded our data analysis in a theoretical framework to reinforce the evidence for external validity. Yet, our findings may be seen as a theory based on empirical research (Eisenhardt, 1989), with practical relevance (Sjøberg et al., 2008), and subject to further testing.

To assure the reliability of the research, we built a research protocol and recorded all the steps taken in the research. This manuscript includes appendices with examples of the artifacts we generated in each of these steps.

5. Data analysis

This section first presents the analysis of data collected in each team and, later, the verification of the propositions considering the cross-case results. The within-case data analysis section comprises the description of each team's: business context, analysis of the evolvement of agile practices, ambidexterity data and project success perception. The cross-case data analysis section shows how the cross-case results support – or not – our propositions.

5.1. Within-case data analysis

Company A. The team in Company A had an ad hoc software development process before agile adoption, which evidences that the agile method was applied to help organizing the development processes. The main characteristic we could perceive in their business is that they have frequent unplanned requirements, but they have to commit to long-term deliveries. They usually plan for three months, although sprints are usually one-month long. They have one product, sold to a number of customers, with little customization. In their daily work dynamics, they have no commitment to follow established processes, so they easily change the way they perform their job. They use tools to support management and development, and experiment new technologies whenever they feel it necessary.

The agile evolvement analysis showed that the team in Company A started pursuing the outcomes listed in the left-hand box in Fig. 4. They used the agile method to help sensemaking the 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 et al., 2005). Sensemaking is a process of organizing, but people "make plausible sense retrospectively, while enacting more or less order into [...] ongoing circumstances" (Weick et al., 2005, p. 409). This team was created with experienced people, so they started with a confident team. Another characteristic of this past experience was the initial understanding of customer demands, and also, the initiatives for the customer to know the team. All the outcomes pursued in the past, and their related evidences – i.e., the practices in the team that led us to infer the pursued outcome – are listed in Table 3.

The middle box in Fig. 4 shows how the outcomes evolved. The confident team, for example, evolved to an assertive team, which is a team that strongly influences decisions. They started being aware of their failures and implemented some initiatives to be able to make "down-to-earth" decisions. The pursuit for high-level source code in the past evolved to a pursuit for high-level delivered software. The customer evolved from being aware of the team to a customer that is confident about the team. For the future (right-hand box in Fig. 4), the team mentioned implementing practices that would allow them to no longer delay deliveries (Defined frequent deliveries), practices to automate tests (Efficient coding) and practices to make the customer more dependent on the team's decisions and positions. These and the other pursued outcomes, and their related evidences, are presented in Table 3.

Table 4 shows the data for the ambidexterity analysis. It presents the percentage of responses in disagreement, agreement or neutral to each of the aspects we analyzed (performance, good alignment, bad alignment, adaptability, bridging ties and strong ties), as well as the *p*-value for the Chi-Square tests. For this team, all the six null hypotheses were rejected, which evidences the team agrees with the perception of ambidexterity. The analysis of the percentage of responses of agreement and disagreement also shows the team identifies ambidexterity abilities in their practices.

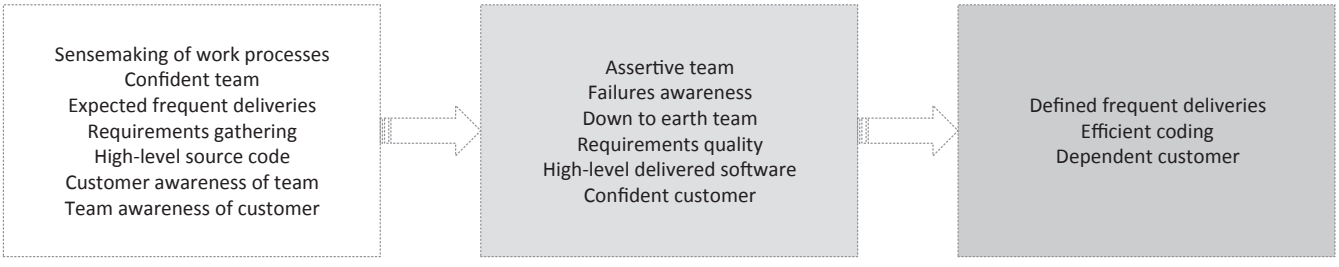

Fig. 4. Agile evolvement analysis for Company A.

Table 3
Evidences for the outcomes in Company A.

Moment	Outcome	Evidences
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; closely knowing each person in the team. Assigning the tasks to the team, but having the members define tasks 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 the customers' needs and their demands.
Present	Assertive team	Defining policies for the acceptance of unplanned requirements; maintaining a sustainable work pace; feeling secure about delivering the software; having the team be able to change task assignments.
	Failures awareness	Considering time in the sprint for debugging, performing functional and unit tests; looking for improvement in the software quality.
	Down-to-earth decisions	Drawing up thorough plans, reporting work status, planning longer sprints, adopting simple metrics.
	Requirements quality	Improving requirements definition (using recorded videos with 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.
	Dependent customer	Hypothesizing customers' needs; defining the roles and rights of the information technology department.

Table 4
Company A ambidexterity data.

	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 5
Projects success perception in Company A.

	Company A	Other teams' means	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

Finally, as suggested by Leppänen (2013), maturity studies should include a means of measuring the success of the projects performed by the team. In Company A, the perception of success is shown in Table 5. We see that there is a perception that the agile method contributed to all benefits, with the exception of Improved Business Process. In comparison with the means of responses from the other teams, the project success was higher in three out of the five success factors.

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

Company B – Team 1. This team adopted agile methods to meet a top-management decision. Before that, they had an ad hoc software development process. They are an inter-disciplinary team, as they develop educational software. Their customers are unknown (they respond to government public demands), so the company has an infrastructure for customers relationship (phone and web contact) and the support of the delivered software is performed by other teams. As they develop under demand, they have a wide variety of projects and products: each project implements a completely new business, with new technology, for different target users. In their daily working process, they change (or even abandon) the established processes as needed and use tools to support management and development.

Fig. 5 shows the evolvement of the pursued outcomes in this team. They started using agile methods with the need to learn them; the sensemaking of work processes came after learning was established. They lack the need for frequent deliveries. As the user is unknown, the product is only delivered at the end of the project through the marketing department of the company. Thus, their iterations focus on finishing code – we called it "Expected frequent finished coding". They had a confident team that evolved to an assertive team. They did not focus on the source-code quality; they just implemented practices to have a high-level delivered software. This team has a specific characteristic that, in the future, they may not use agile methods. They were about to go through an organizational reengineering that would separate teams by roles and the customer would be represented by a new department. This is what we called Specialist Team and Represented Customer in the right-hand box in Fig. 5. These outcomes and the others identified for Company B – Team 1 are shown in Table 6 along with their evidences.

The ambidexterity data analysis for Team 1 in Company B showed that the team does not agree in the perception of good alignment and adaptability in management (the null hypothesis could not be rejected). This means that the team feels like receiving conflicting objectives, as well as feels the lack of flexibility to meet customer needs (Table A.1 in Appendix A shows the statements evaluated in each of the ambidexterity aspects). Table 7 shows the percentage of responses in disagreement, agreement or neutral to each of the aspects and the *p*-value for the Chi-Square tests. Analyzing the percentages of agreement and disagreement, one can verify that part of the team disagree with a team's good performance. According to the team leader, in the validation presentation, these bad perceptions are due to the

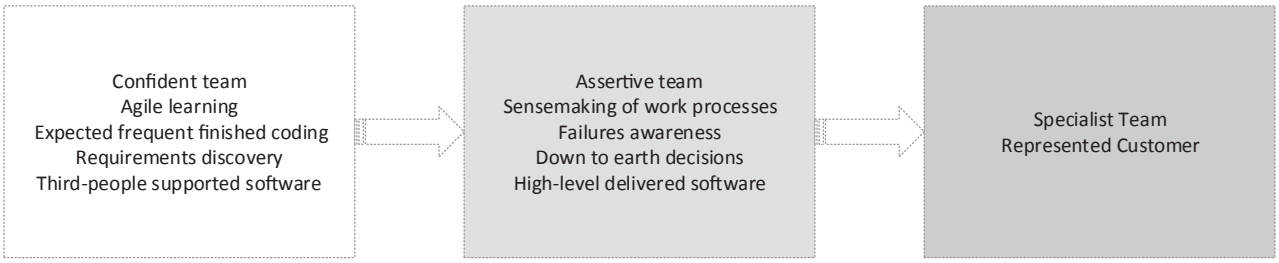

Fig. 5. Agile evolvement analysis for Company B – Team 1.

Table 6
Evidences for the outcomes in Company B – Team 1.

Moment	Outcome	Evidences
Past	Confident team	Creating an experienced team; closely knowing each person in the team; 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 finishes.
	Third-people supportable software	Creating documentation at the end of the process; using text documents to define requirements.
Present	Assertive team	Letting the team 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.
	Failures awareness	Considering time in the sprint for debugging.
	Down-to-earth team	Drawing up thorough plans; defining requirements iteratively.
	High-level delivered software	Performing unit and functional tests; applying tools to support the development.
Future	Specialist team	Separating the team into one team for each role; formalizing software architecture definition; having a team perform functional tests formally.
	Represented customer	Have a formal structure to represent customer.

Table 7
Company B – Team 1 ambidexterity data.

	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

Table 8
Project success perception in Company B – Team 1.

	Company B – Team 1	Other teams' means	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

rigid structure of the company as a whole, and not regarding the team itself.

When measuring the projects success perception in Company B – Team 1, we got the data shown in Table 7. This table also shows the mean perception for the other teams studied and the standard deviation. The evaluation of all the success factors in this team was higher than the mean of the other teams' responses.

We conclude that Company B – Team 1 lack the need of delivering 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 projects success perception is good, the team does not present an agreement on ambidextrous abilities.

Company B – Team 2. The context before agile adoption was an ad hoc software development process, with a top-down agile adoption initiative. Although this is the same company as Team 1, the products they develop are completely different. This is a new product development team, which creates its own requirements. Their inter-disciplinary skills contribute to new ideas to emerge. They use simple tools to support management and development; and adapt the development process whenever needed.

The agile evolvement analysis in Fig. 6 shows that the team started using agile methods with focus on agile learning, which later allowed them to make sense of their processes. In the past, their iterations were used to control the coding; but it evolved to iterations to deliver working software. They had a specialist team, with roles separated into different groups, which evolved to the agile characteristic of a single team with multiple roles. This team started the agile adoption with a responsive characteristic and evolved to a confident team that is currently an assertive one. In the past, they did not focus on high-level source code, but started focusing on that in the present. The practices to support "down-to-earth" decisions were also added later. In the future, the team plans to look for requirements quality and high-level delivered software. The strategy they plan to improve requirements quality is to use system analysis diagrams in requirements definition. These and the other pursued outcomes are presented in Fig. 6 and Table 9 presents the evidences that support the outcomes in this case.

The analysis of the ambidexterity perception of the team did not reject the null hypothesis for good alignment, bad alignment and adaptability, as shown in Table 10. It means that there is not a consolidated perception in the team about the management alignment, conflicting objectives and adaptability.

The perception of the project success factors in Company B – Team 2 is shown in Table 11. In comparison with the other teams means, they had higher or the same perception for three out of the five factors.

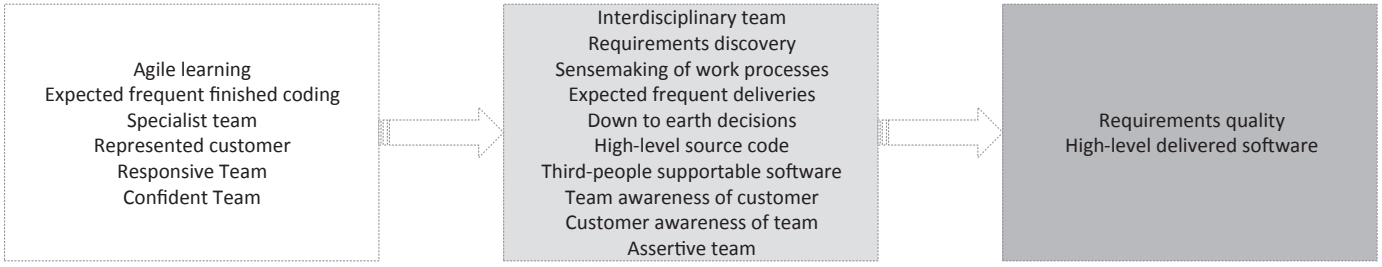

Fig. 6. Agile evolvement analysis for Company B – Team 2.

Table 9
Evidences for the outcomes in Company B – Team 2.

Moment	Outcome	Evidences
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 functional tests performed in a separate team; having the roles separated into 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 know requirements before starting to 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.
	Down-to-earth decisions	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 help defining 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.

Table 10
Company B – Team 2 ambidexterity data.

	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

Table 11
Project success perception in Company B – Team 2.

	Company B – Team 2	Other teams' means	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

In sum, Team 2 in Company B started an 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, which became confident and, currently they are active actors in the projects. The initiatives to have the customer understand the team and vice-versa were not implemented in the beginning of agile adoption, but in a later stage. The ambidexterity analysis did not yield the results expected, but the projects success has been identified mainly by reduced delivery schedules.

Company C. The context for agile adoption in this team is quite different from the other teams. They used to have a strong and defined traditional development process. Their business context is mainly characterized by very strict delivery dates. They develop great customizations for various software platforms that comprise the information technology infrastructure for telecommunication services. As it is a big company, the roles are separated into different hierarchical structures. To develop the software projects, people are temporarily placed physically together. The procedures in the software development are rigid, so little space is left for process adaptation; however, the team leader emphasized that little details change from project to project. To support daily activities, they have a strong infrastructure of tools to support management and development.

The agile evolvement analysis shows that agile adoption started with the sensemaking of work processes. The iterations were initially implemented to control the coding process, which later evolved to expected frequent deliverables. These deliverables are not actually delivered because the customizations they perform cannot be delivered in parts. The practices that contributed to "down-to-earth" decisions where performed from the beginning, once they come from a traditional process. They reported a responsive team situation in the present; thus, in the future there should be actions to improve confidence in team members. They feel the lack of standardization of

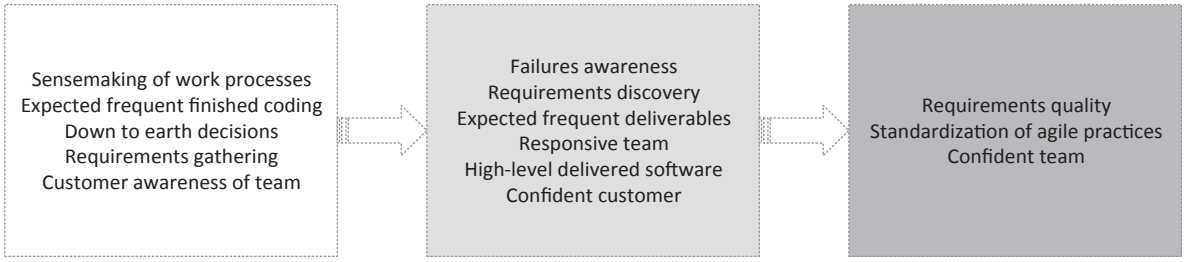

Fig. 7. Agile evolvement analysis for Company C.

Table 12
Evidences for the outcomes in Company C.

Moment	Outcome	Evidences
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.
	Down-to-earth decisions	Developing project scope using work breakdown structure diagrams; starting project only with a perceived scope maturity; reporting work status; having people help with administrative tasks.
Present	Failures awareness	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 the macro plan is defined by software architects; top management listens to suggestions for improvement, but does not necessarily consider 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.
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.

agile practices across the teams is a problem, and it should also be the focus of initiatives in the future, as well as the quality of the requirements definition. Fig. 7 summarizes the evolvement of the outcomes for Company C and Table 12 presents the evidences we found for each outcome.

The ambidexterity data analysis for Company C team is shown in Table 13. The null hypothesis for good alignment, bad alignment and adaptability were not rejected for this team, which means that there is not a consolidated perception of the team about the alignment to objectives, conflicting objectives and adaptability. In the validation presentation, the team leader showed to be concerned about the 83.3% of neutral opinion on the good alignment investigation. The issue was taken to top management, who justified it as a problem with cultural change and resistance to agile methods, the 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.

With respect to project success perception, we verified that Company C responses were all lower than the mean of the other teams' responses, with the exception of the reduced delivery schedules. They still feel a poor ability to meet changing customer requirements (Table 14).

The investigation of the team in Company C showed a scenario where a cultural change was necessary for agile adoption. The team started pursuing the sensemaking of work processes, with 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 in the bad results of the ambidexterity analysis. From all the teams we analyzed, this was the one with the lowest perception of projects success.

Table 14
Project success perception in Company C.

	Company C	Other teams' means	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

5.2. Cross-case analysis

Based on the analysis of the four individual cases and on the patterns that emerged in the cross-case analysis, we are able to evaluate how their empirical data support or not the propositions of the study.

Table 13
Company C ambidexterity data.

	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

Our first proposition stated that *the team plays a central role in agile software development maturity*. To evaluate this proposition, we verified which position of the teams was in each case. Company A, with the best results for ambidexterity and project success, had a team that started working with agile methods feeling confident. They were experienced people and evolved to an assertive team. Company C, with the worst results for ambidexterity and project success, reported to have problems with the team. It was characterized by responsive people that resisted engaging in the self-organizing behavior of confident teams. Company B – Team 2 reported to initially have problems with a responsive team and, to evolve in agile adoption, the members that did not engage in the process had to leave the team.

We also lack data that show the processes definition and standardization growing in the agile practices evolvement. Instead, teams apply some practices to make their decisions more grounded (e.g. project tracking, report work status etc.) but evolvement is perceived in their behavior, and also in their relationship with the customer.

Another evidence is to analyze the time the team has worked with agile methods. Company A, less experienced with agile methods – but the one with the focus on experienced people – had good results with ambidexterity and project success. The other teams, with longer use of agile methods – but with less focus on creating a high-level team – had worse results in ambidexterity or project success. We thus conclude we have evidence to confirm our first proposition.

The 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. First considering that maturity is a state of completeness (Maier et al., 2012) and that maturity should be related to the success of the projects (Leppänen, 2013), we show in Table 15 the means for project success perception in each team and the number of null hypothesis not rejected in the ambidexterity analysis.

Team 1 in Company B had the highest mean in project success, but two null hypotheses were rejected. Thus, we consider Company A the one with the best result: a project success perception mean close to the highest, but none of the null hypotheses was rejected. On the other hand, Company C has the lowest project success perception and, also, three null hypotheses not rejected. These relations evidence that ambidexterity is related to project success and, thus, to maturity in agile software development. This confirms our second proposition.

Moreover, the experimentation processes (see point 2 in Fig. 1) were present in all teams. The openness to experiment new procedures, tools and ways to work was very evident in Company A, Company B – Team 1 and Company B – Team 2. The one we lack the evidence for experimentation culture was the team in Company C.

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 in each team. They started their agile adoption differently and the practices also evolved differently. We could, however, conclude that the pursued outcomes are actually similar, and a pattern could be identified in the progression of these outcomes. The variety of practices also shows the exploration of the teams, experimenting different ways to get to outcomes, which reinforces our perception of the ambidexterity abilities in the maturing process.

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

Moreover, as a pattern in the outcomes emerged from the comparison of the cases, our cross-case analysis uncovered six categories of pursued outcomes and how they evolved in real agile teams (Fig. 8): practices, team, deliveries, requirements, product and customer. We named it the *Progressive Outcomes* framework for agile software development. Tables 3, 6, 9 and 12 show the practices the teams participating in this study implemented to pursue the outcomes described in this section. One should argue that not all the outcomes identified in the cases are included in this framework. Indeed, the pursued outcomes Standardization of agile practices, Specialist team, Interdisciplinary team, Represented Customer, and Third-people supportable software were specific to the context and the moment a specific team was going through and, for this reason, were not included in the general framework.

The *practices* category comprises the outcomes the teams pursue when they decide to change the way they work. In agile software development adoption, it starts with initiatives for agile learning. The teams implement the agile method "by the book" and it evolves to a sensemaking of the work processes. This process includes taking the method learned and tailoring it to particular needs, according to specific past experiences. Later, the teams start to invest in practices that pursue making "down-to-earth" decisions, which includes, for example, using tools to track the process, having the team report work status and simple metrics. This dynamics is similar to the three levels of practice introduced by Cockburn (2007) as the Shu-Ha-Ri distinction.

The *team* 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 management position of command and control. This team may evolve to a confident team, in which team members start positioning themselves in the decisions and, later, the assertive team is the one whose members are active voices in the project and in the process improvement initiatives.

The *deliveries* category describes how the pursued outcomes for deliveries evolve. Teams start investing on 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. In the teams included in this study, this initiative was related to implementing a functional test phase that would assure the delivery is ready, but will not be delivered yet. This process of producing ready deliverables then evolves to actual deliveries at the end of the iterations – usually late deliveries. In a last step, the team starts working on practices to have a defined delivery, one that has a decrease in delay.

The *requirements* Progressive Outcomes start with requirements gathering, a process of eliciting requirements close to the traditional software process, with most 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 requirements elicitation, to use stories, and requirements are allowed to change. The last outcome they pursue is to improve the quality of the requirements to

Table 15
Relationship of the perceptions of project success and of ambidexterity.

	Company A	Company B – Team 1	Company B – Team 2	Company C
Mean for project success perception	4.2	4.6	3.8	3.4
Number of null hypothesis not rejected in the ambidexterity analysis	0	2	3	3

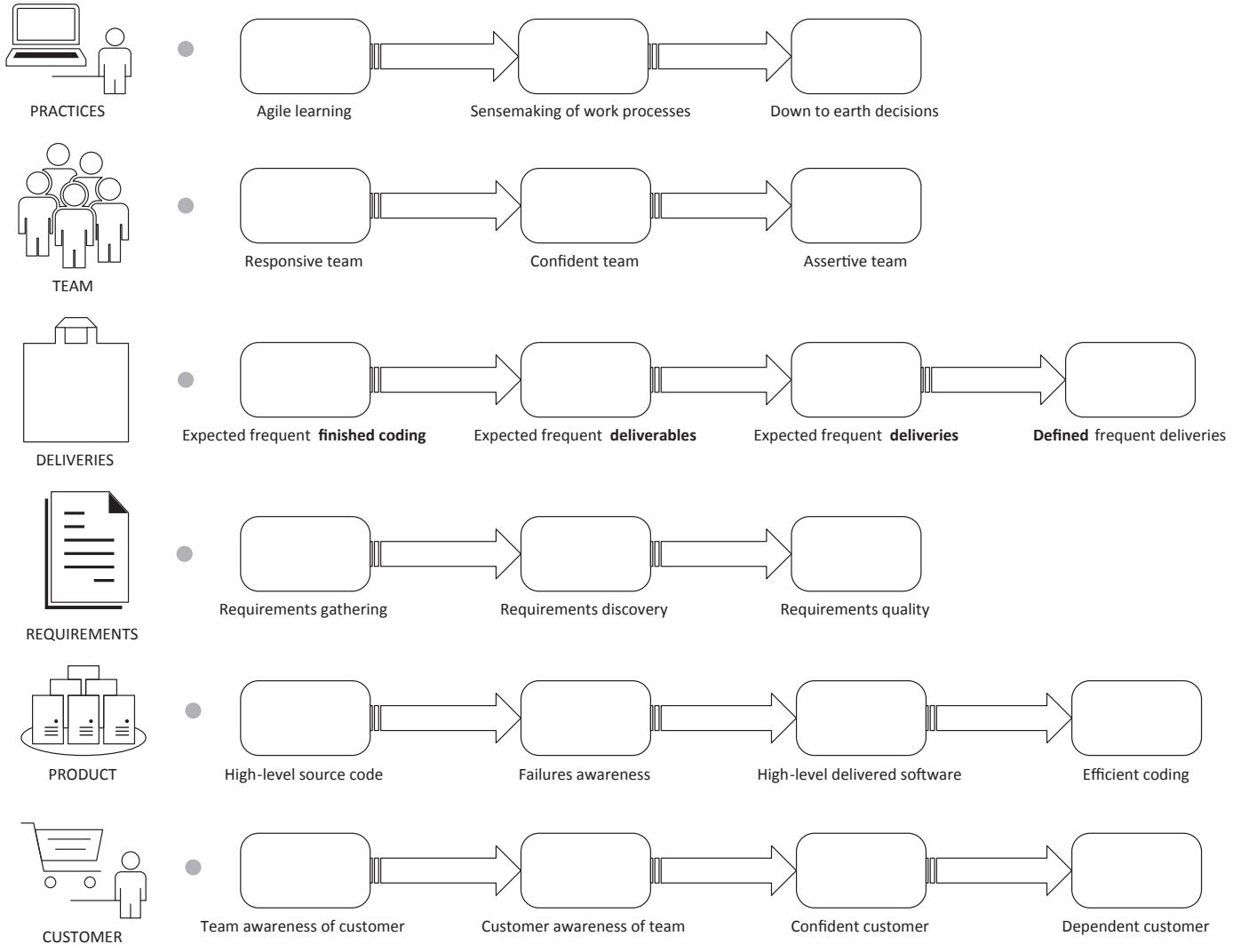

Fig. 8. The Progressive Outcomes framework for agile software development maturity.

make sure they meet customers' expectations. Quite different practices appeared to accomplish this outcome, such as using videos to record customer requirements, or using systems analysis diagrams.

The outcomes for the *product* category describe what the team pursues when implementing 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 "failures awareness" because it is when the team realizes it delivers bugs, that it has to fix them and that it has to adjust the processes to accomplish that. After having a good code and the awareness of the failures, the focus is to have a good delivered product. Then, a next step is to focus on a high-level delivered software: they invest in having a functional tester, or a test team, for example. The last outcome is efficient coding, when practices such as test automation are implemented to increase efficiency in the coding process, sustaining the quality that was already obtained with the outcomes that were accomplished previously.

The last category, the *customer*, comprises the outcomes the team pursues when implementing practices to improve relationship with the customer. They start with the team getting aware of the customer, understanding the customer's processes and needs. Then, when the relationship evolves, the customer gets aware of the team, knowing 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 hence becomes dependent on the team because there is so much confidence that the team helps defining requirements and solutions for the customer's business problems.

This framework represents a discontinuous process of agile software development evolvement. In the validation presentation we did with the team leaders, we presented which of the outcomes each team had accomplished (in a subjective assessment) and it was clear that each team evolves in the outcomes that are important to their business contexts. For example, Company A did not invest in "Agile Learning", as they hired an experienced team. Company C had "Down-to-earth decisions" before agile adoption. Company B – Team 1 did not pursue any of the customers' outcomes, as they did not have this need.

This section presented the results from the data analysis. We presented the findings for each case individually and the evaluation of the propositions of the study, based on the cross-case analysis. The next section discusses these results and their relevance to practice.

6. Discussion

This study aimed to identify how agile software development teams evolve to maturity. We identified it is a discontinuous process of experimentation, based on Progressive Outcomes, which the team pursues through the implementation of practices that cannot

be prescribed. The team implements practices to chase outcomes in work practice, in its own behavior, in the deliveries, in the requirements, in the final product and in the relationship with the customer. It is a framework that reflects an ambidextrous work experience: the team should be aligned with specific outcomes – exploitation, but free to adapt practices as they please – exploration.

Our findings have shown that the team plays a central role in the agile software development maturing process. When we consider that people play this main role, the processes come in secondary place (Fontana et al., 2014). The implication of this finding is that the current models for maturity in software development and their requirements for assessment cannot be applied to our framework. The majority of maturity models, in the software engineering field and elsewhere, are based on process definition and control (Maier et al., 2012). However, people are the focus in any software development initiative and they are particularly important in agile methods (McHugh et al., 2012). In the software process improvement efforts, management should consider the agile software development team as a self-organizing system (Hoda et al., 2012) and should focus on empowering the team, not on implementing control tools (Middleton and Joyce, 2012). The use of simple rules to drive the work is an example of how the management may foster the emergent behavior in agile teams (Power, 2014).

This central role teams play show the importance of the "Team" outcomes evolvement in the framework we propose (Fig. 8). The overall agile software development processes is conducted by the team and the way it evolves directly reflects in how processes are performed (Kettunen, 2014). Our study has shown, for example, that responsive teams define their on tasks, but in a personal level. Estimates and general decisions are made by software architects or by the Scrum Master. They may suggest improvements, but these require approval from top management and, thus, do not create confidence in the team. It leads the team to not taking responsibility. Confident teams are those who take more general decisions – which influence more the project – as estimates and priorities. In these teams people know each other closely and there is mutual trust. When evolving to an assertive team, team members can change their assignments, they improve their own work processes and feel secure to define politics for unplanned requirements and, thus, protecting the team from, for example, extra requirements during the sprint. We consider that the three processes of team development described by Kuipers and Stoker (2009) – task managing, internal relations, external relations and improvement – would be performed throughout this evolvement in the team as well as in the relationship with customer.

Although Kuipers and Stoker (2009) say that "team development should not be regarded as a goal in itself [. . .], i.e. a team is a mean to an end rather than the end in itself" (Kuipers and Stoker, 2009, p. 408), our findings are in accordance with Kettunen (2014) when he suggests that improvements in agile software development may be addressed by improving team performance. Current literature provides a number of suggestions on how to improve teamwork and, we believe, aid on the pursue of new outcomes. Fostering emergent behavior, for example, may be done by defining simple rules, instead of prescribing practices (Power, 2014). Self-managed teams can be developed by avoiding team members with a high degree of individual autonomy and a Scrum Master focused too much on command and control (Moe et al., 2010). Other factors that also negatively influence the creation of a self-managed team are: little feedback and knowledge backup, increasingly specialization of team members, lack of trust among team members and lack of a shared mental model of what the outcome of the project should be (Moe et al., 2010). Removing impediments to flow in work is also a way to improve processes, according to Power and Conboy (2014). On team, it could be done by reducing handovers, context switching, unnecessary motion and unmet human potential. Choices of the team design are also an important factor that impact team productivity. Melo et al. (2013) identified that full time team members contribute to team focus; mixed team members, as novice members contribute flexibility and experienced ones contribute knowledge; small teams, which lead to better communication, conflict management, commitment and sense of responsibility; and team collocation, which helps in negotiation and planning of requirements.

This team matures in agile software development by developing ambidextrous abilities – another evidence in our findings. Studies in the agile software development field have shown benefits in combining these dual forces of exploration and exploitation (March, 1991). For example, the importance of preserving disciplined practices when adopting agile methods (Boehm and Turner, 2004), as well as the established trend to combine plan-driven and agile characteristics in a single project (Baskerville et al., 2011). The synchronization of exploration and exploitation has already been observed in agile teams dynamics (Vidgen and Wang, 2009), and, likewise, ambidexterity (Ramesh et al., 2012). Even in the behavior of agile practitioners, a duality has been perceived: serious professionals with a free-spirit and joking behavior (Hazzan and Leron, 2010). The preference for exploitation activities, with codified routines, hinders the dynamic capabilities of the organization (Eisenhardt and Martin, 2000). This dynamic capability, in software development process improvements, is characterized by the "ability to improve software development process with respect to changing circumstances" (Clarke and O'Connor, 2011, p. 29).

Hence, in this study, we added the perspective of developing the ambidextrous ability to mature in agile software development, which is a challenge, given that there is not such a unique recipe for ambidexterity (Gibson and Birkinshaw, 2004). Our proposal is thus fostering alignment (exploitation) with clear expected outcomes, but leaving space for the emergence of variety (exploration) by not prescribing the practices the team should implement.

This space for exploration seems essential in the maturing process since the lack of pattern in the adoption of the practices was another finding in the cases we investigated. There were, already, some clues about this in the existing studies in the agile software development maturity field. In the model proposed by Sidky et al. (2007), for example, a set of practices was defined for each maturity level but, in the validation of the framework, practitioners pointed out that there is a need for tailoring and considering the experiences of the organizations. Kettunen (2012) got to a similar conclusion, as well as the survey performed by Schweigert et al. (2012). Our previous survey (Fontana et al., 2014b) indicated that agile practitioners see value in following a maturity model, only if space is left for tailoring. It confirms that the trend of tailoring agile methods is already established (Bustard et al., 2013) as agile teams focus on having the job done, and not on following specified processes (Adolph et al., 2012; Coleman and O'Connor, 2008). The one-size-fits-all approach is inappropriate for agile software development (Armbrust and Rombach, 2011; Sheffield and Lemétayer, 2013). We confirmed this variety of practices in our study and considered the need for allowing context-specific practices in our framework.

In the development of dynamic capabilities, Eisenhardt and Martin (2000) observed that they imply equifinality – multiple starting points to develop the same capability. Similarly, agile software development maturing process does not seem to be comprised of a linear sequence of adoption of practices. It actually seems to be a discontinuous process, in which the team chases the outcomes simultaneously for the improvements in the work practice, in their own behavior, in the deliveries, in the requirements, in the final product and in the relationship with the customer. The Progressive Outcomes framework proposes a "semi-structure" for the maturing process, contrasting with the current agile maturity models (Benefield, 2010; Lui and Chan, 2005; Nawrocki et al., 2001; Patel and Ramachandran, 2009; Qumer and Henderson-Sellers, 2008; Sidky et al., 2007; Yin et al., 2011): 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 and 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" (Allison and Merali, 2007, p. 678). This intended vision is what we named pursued outcomes.

These findings, thus, uncover the subjectivity involved in the assessment of agile practices. How can a maturity model be useful if it lacks the support to assess the team current stage? Maier et al. (2012) already pointed out that maybe organizational maturity is more subjective than we are used to. The assessment in agile methods cannot be based on extensive documentation (Nawrocki et al., 2001) and further research should investigate how to assess if and how the Progressive Outcomes have been accomplished in agile teams. Nevertheless, we believe our approach answers the claim that "an agile approach to software process improvement would be responsive and flexible to local needs, encourage innovation in the process, build software process improvement projects around those who are motivated, encourage self-organizing competent teams, and promote sustainable development of the process" (Allison and Merali, 2007, p. 679).

7. Conclusion

This study presented a framework for maturing in agile software development. We built this framework based on the analysis of qualitative and quantitative data in four Brazilian agile teams. Our theoretical foundation, based on complex adaptive systems theory, led us to build a framework for agile software evolvement that considers people as agents who play the key role in the maturing process, sees ambidexterity as a key ability to maturity, and does not prescribe practices, but describes outcomes teams actually pursue. Agile maturity comes from a discontinuous process of pursuing Progressive Outcomes in the practices, in the team, in the way deliveries are performed, in the way requirements are defined, in the quality of the final product and in the customer relationship.

A limitation of this study is that the conclusions are based on a four-case study. Although researchers in qualitative studies cannot pursue statistical generalization (Eisenhardt, 1989), our results are specific to the contexts and the profiles of the teams included in the study. Our future work considers analyzing more cases to replicate and to update these findings.

Considering the generalizability limits, we have proposed a framework based on a theory for agile software development maturity, confirmed through empirical data. The main implication for research is the challenge of uncovering subjective ways to assess the agile teams and favor the application of such framework as a practical software process improvement guide. For practitioners, our results address a concern agilists have about following prescribed models. In comparison with the other agile improvement guides available in industry and research, our framework brings a novel approach: sustaining agile values in the maturing process and allowing the emergence of context-specific practices.

Researchers in the field might consider further research on enabling the framework to become applicable in industry. We suggest, thus, some research questions that could be addressed:

- How the proposed framework could comply with international standards for process measurement frameworks, such as ISO/IEC 33003 (ISO/IEC, 2014)?
- Which are the assessment approaches that enable evaluation of the maturity of an agile team in each of the categories of the Progressive Outcomes framework? The structure suggested by the Test Process Improvement model (Andersin, 2004) might be a reference for this research question.
- Are there "common paths" for agile improvement, even considering the variety of contexts where agile teams have been performing their jobs? Although we still lack evidence that maturity stages are applicable, there is a possibility that maturity paths could be defined for different agile software development contexts.

Acknowledgments

We would like to thank the team leaders and the practitioners for telling us their stories and answering our questionnaires.

Appendix A

Table A.1
Questions for ambidexterity evaluation and reference authors.

Reference author	Aspect	Question
Gibson and 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-proposals 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 are flexible enough to allow us to respond quickly to changes in our markets
		The management system in this organization evolve 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 each other's
	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 between members

Appendix B

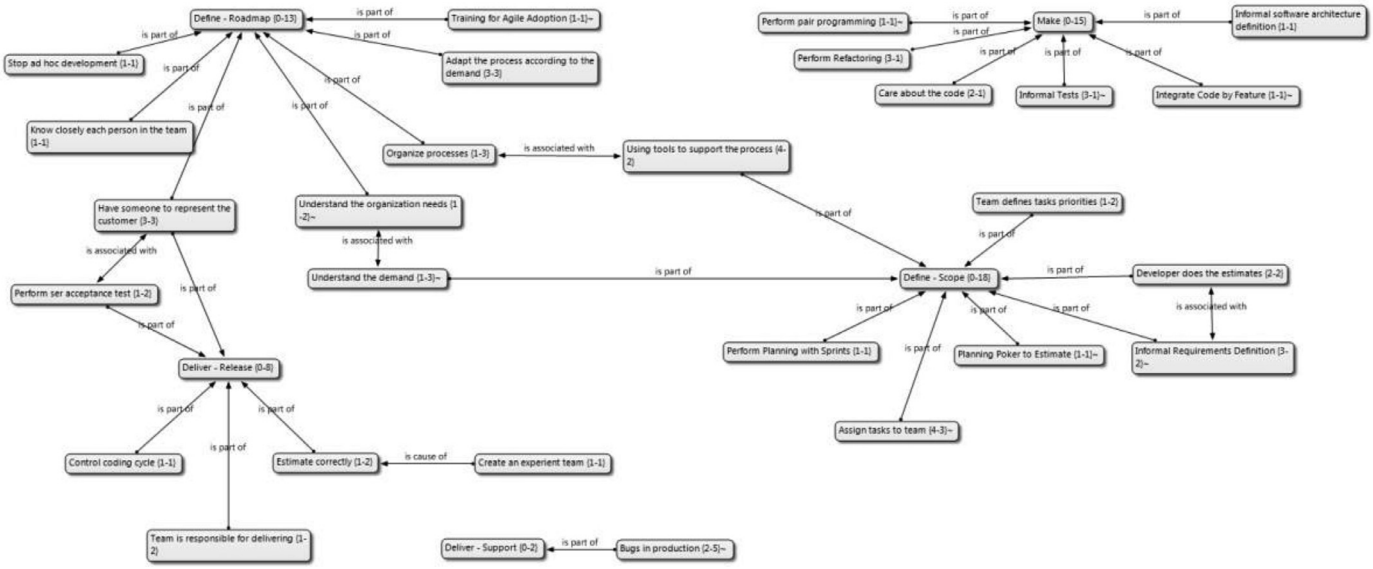

Fig. B.1. Example of code network for Company A – Past.

Appendix C

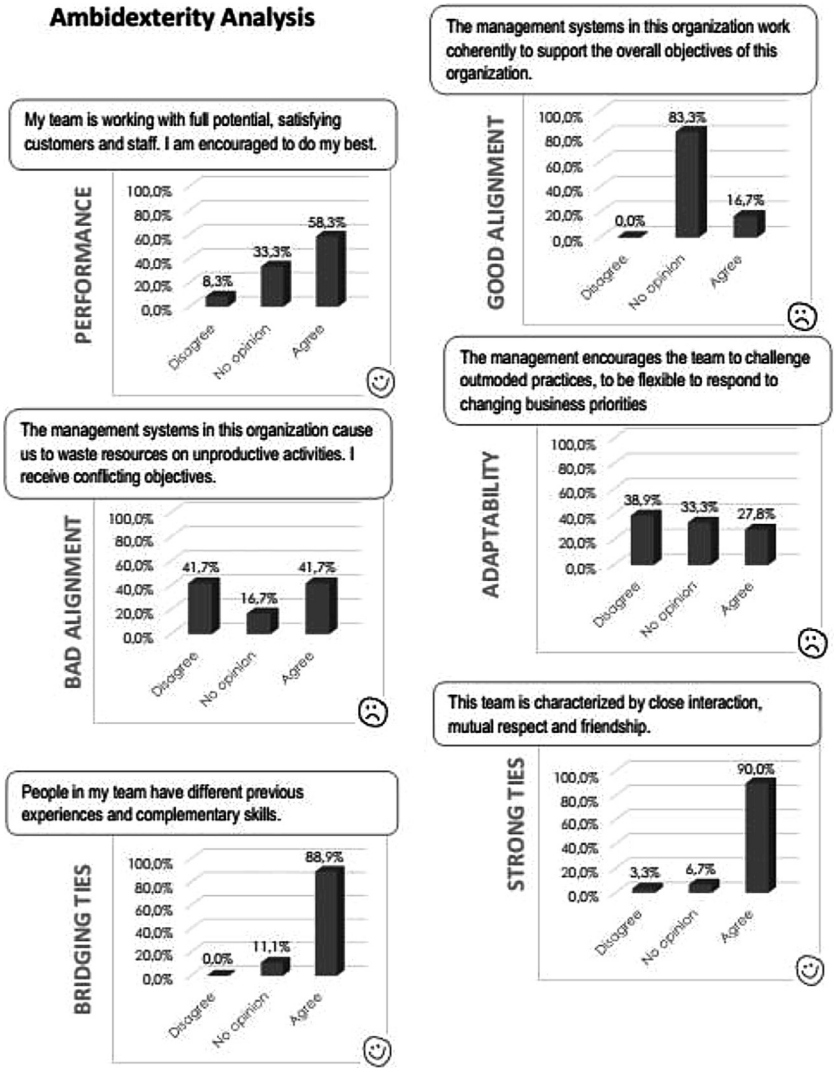

Fig. C.1. Ambidexterity results in the Case Report. Example for Company C.

Appendix D

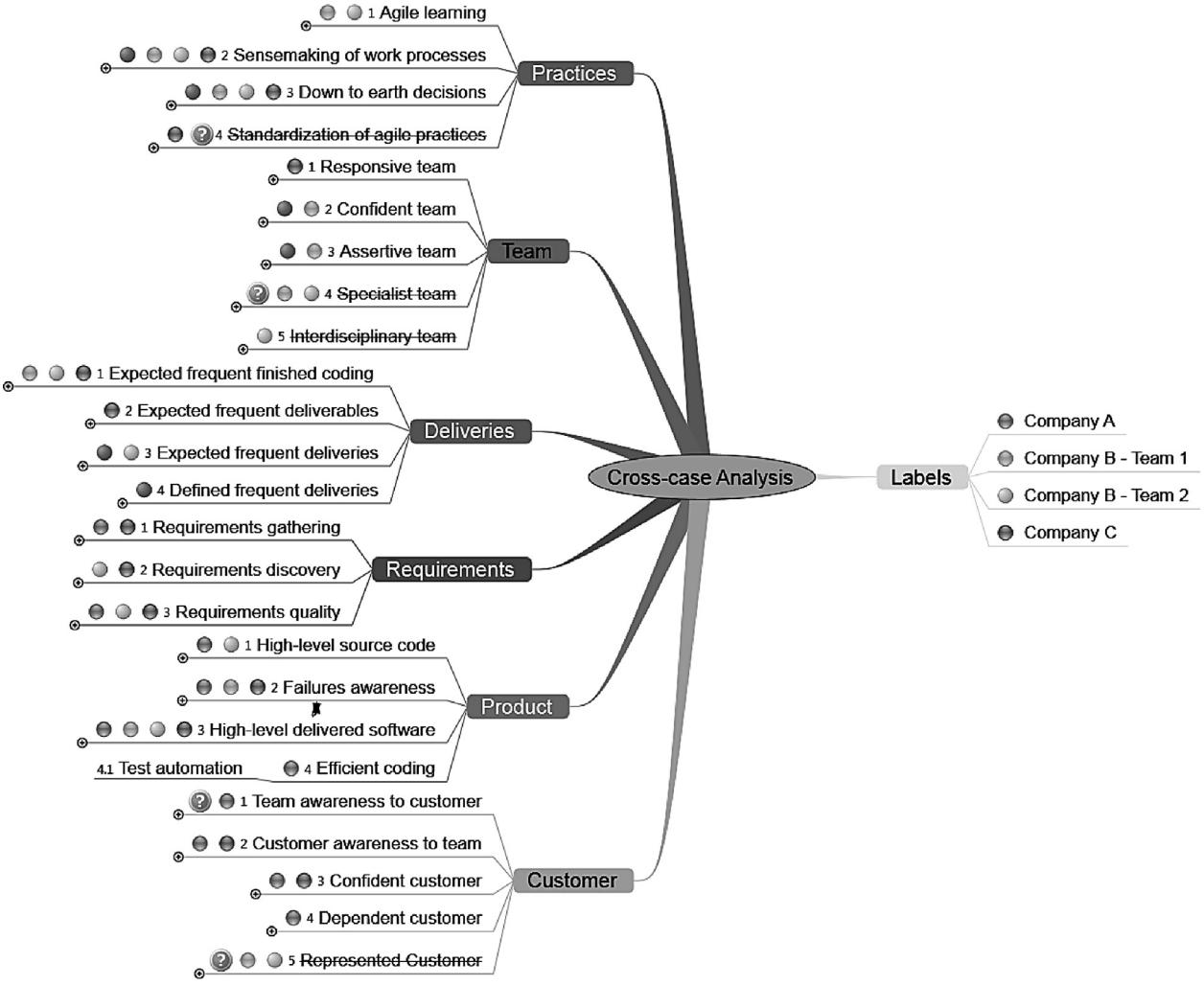

Fig. D.1. Cross-case analysis mind map.

References

Adolph, S., Krutchen, P., Hall, W., 2012. Reconciling perspectives: a grounded theory of how people manage the process of software development. J. Syst. Softw. 85 (6), 1269–1286. doi:10.1016/j.jss.2012.01.059.

Allison, I., Merali, Y., 2007. Software process improvement as an emergent change: a structurational analysis. Inf. Softw. Technol. 49 (6), 668–681. doi:10.1016/j.infsof.2007.02.003.

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 Comp. Sci. 3, 893–897. doi:10.1016/j.procs.2010.12.146.

Andersin, J., 2004. TPI – A Model for Test Process Improvement. Seminar on Quality Models for Software Engineering. Helsinki http://goo.gl/9JSFUQ.

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 2005, pp. 193–201. doi:10.1109/ADC.2005.42.

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. New York, NY, USA, pp. 237–241. doi:10.1145/1987875.1987920.

Augustine, S., Payne, B., Sencindiver, F., Woodcock, S., 2005. Agile project management: steering from the edges. Commun. ACM 48 (12), 85–89. doi:10.1145/1101779.1101781.

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, L., 2011. Análise de Conteúdo. Edições 70.

Baskerville, R., Pries-Heje, J., Madsen, S., 2011. Post-agility: what follows a decade of agility? Inf. Softw. Technol. 53 (5), 543–555. doi:10.1016/j.infsof.2010.10.010

Beck, K. et al., 2001. Agile Manifesto. Available in http://agilemanifesto.org/. Accessed in 2014, August.

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. Honolulu, HI, pp. 1–7. doi:10.1109/HICSS.2010.337.

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. 718–729. doi:10.1109/ICSE.2004.1317503.

Buglione, L., 2011. Light maturity models (LMM): an agile application. Profes '11: Proceedings of the 12th International Conference on Product Focused Software Development and Process Improvement, pp. 57–61. doi:10.1145/2181101.2181115.

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). April 22–24, pp. 139–146. doi:10.1109/ECBS.2013.11.

Caffery, F.M., Pikkarainen, M., Richardson, I., 2008. AHAA – agile, hybrid assessment method for automotive, safety critical SMEs. ICSE '08: Proceedings of the 30th ACM/IEEE International Conference on Software Engineering. 10–18 May, pp. 551–560. doi:10.1145/1368088.1368164.

Campbell-Hunt, C., 2007. Complexity in practice. Hum. Relat. 60 (5), 793–823. doi:10.1177/0018726707079202.

Clarke, P., O'Connor, R.V., 2011. An approach to evaluating software process adaptation. In: O'Connor, R.V. (Ed.). Software Process Improvement and Capability Determination: 11th International Conference, SPICE 2011, Dublin, Ireland, May 30–June 1, 2011. Proceedings, pp. 28–41. doi:10.1007/978-3-642-21233-8_3.

CMMI Product Team, 2010. CMMI for Development, Version 1.3. Software Engineering Institute, Carnegie Mellon University, Pittsburgh, Pennsylvania, Technical Report CMU/SEI-2010-TR-033. Available at http://goo.gl/kJzxiy.

Cockburn, A., 2007. Agile Software Development: The Cooperative Game, second edition Addison-Wesley, Boston.

Cohan, S., Glazer, H., 2009. An agile development team's quest for CMMI maturity level 5. Agile Conference. 24–29 August, pp. 201–206. doi:10.1109/AGILE.2009.24.
Coleman, G., O'Connor, R., 2008. Investigating software process in practice: a grounded theory perspective. J. Syst. Softw. 81 (5), 772–784. doi:10.1016/j.jss.2007.07.027.
Conboy, K., Coyle, S., Wang, X., Pikkarainen, M., 2011. People over process: key challenges in agile development. IEEE Softw. 28 (4), 48–57. doi:10.1109/MS.2010.132.
Eijnatten, F.M., 2003. Chaordic systems thinking: chaos and complexity to explain human performance management. In: Putnik, G.D., Gunasekaran, A. (Eds.), Business Excellence 1: Performance Measures, Benchmarking and Best Practices in New Economy. University of Minho Press, Portugal, pp. 3–18.
Eisenhardt, K., 1989. Building theories from case study research. Acad. Manage. Rev. 14 (4), 532–550. doi:10.5465/AMR.1989.4308385.
Eisenhardt, K.M., Martin, J.A., 2000. Dynamic capabilities: what are they? Strateg. Manage. J. 21 (10-11), 1105–1121. doi:10.1002/1097-0266(200010/11)21:10/11<1105::AID-SMJ133>3.0.CO;2-E
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? J. Syst. Software. 97, 140–155. doi:10.1016/j.jss.2014.07.030
Fontana, R.M., Reinehr, S., Malucelli, A., 2014b. Maturing in agile: what is it about? In: Proceedings of the 15th International Conference, XP 2014. Rome, Italy, pp. 94–109. doi:10.1007/978-3-319-06862-6_7
Gibson, C., Birkinshaw, J., 2004. The antecedents, consequences, and mediating role of organizational ambidexterity. Acad. Manage. J. 47 (2), 209–226. doi:10.2307/20159573.
Hazzan, O., Leron, U., 2010. Disciplined and free-spirited: 'time-out behaviour' at the Agile conference. J. Syst. Softw. 83 (11), 2363–2365. doi:10.1016/j.jss.2010.06.018.
Hidalgo, C., 2011. The value in between: organizations as adapting and evolving networks. In: Allen, P., Maguire, S., McKelvey, B. (Eds.), The SAGE Handbook of Complexity and Management. SAGE Publications, London.
Hoda, R., Noble, J., Marshall, S., 2012. Self-organizing roles on agile software development teams. IEEE Trans. Softw. Eng. 39 (3), 422–444. doi:10.1109/TSE.2012.30.
Humphrey, W., 1995. A Discipline for Software Engineering. Addison-Wesley Publishing Company, Reading, MA, 789.
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: 15504-1, 2004. Information Technology – Process Assessment – Part 1: Concepts and Vocabulary. ISO/IEC, Geneva, Switzerland http://goo.gl/DZJfuS.
ISO/IEC: 33003. 2014. Accessed in 2014, October. Information technology – Process assessment – Requirements for process measurement frameworks. Under development at http://goo.gl/LtmPpr.
Jakobsen, C.R., Johnson, K.A., 2008. Mature agile with a twist of CMMI. Agile Conference 2008, 4–8 August, pp. 212-217. doi: 10.1109/Agile.2008.10.
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.
Kettunen, P., 2012. Systematizing software development agility: towards an enterprise capability improvement framework. J. Enterp. Transform. 2 (2), 81–104. doi:10.1080/19488289.2012.664610.
Kirk, D., Tempero, E., 2012. A lightweight framework for describing software practices. J. Syst. Softw. 85 (3), 582–595. doi:10.1016/j.jss.2011.09.024.
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 http://goo.gl/hnw7uh.
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 20 (2), 399–419. doi:10.1080/09585190802670797.
Kurapati, N., Manyam, V.S., Petersen, K., 2012. Agile software development practice adoption survey. In: Wohlin, C. (Ed.). Agile Processes in Software Engineering and Extreme Programming: 13th International Conference, XP 2012, Malmö, Sweden, May 21–25, 2012. Proceedings, pp. 16–30. doi:10.1007/978-3-642-30350-0_2.
Leppänen, M., 2013. A comparative analysis of agile maturity models. In: Pooley, R. (Ed.), 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, Z., Dan, S. 2012. Research on combining scrum with CMMI in small and medium organizations. Proceedings of the International Conference on Computer Science and Electronics Engineering, 23–25 March, Hangzhou. pp. 554–557. doi:10.1109/ICCSEE.2012.477.
Lui, K.M., Chan, K.C.C., 2005. A road map for implementing extreme programming. In: Li, M., Boehm, B., Osterweil, L.J. (Eds.). Unifying the Software Process Spectrum: International Software Process Workshop, SPW 2005. Beijing, China, pp. 474–481. doi:10.1007/11608035_38.
Lukasiewicz, K., Miler, J., 2012. Improving agility and discipline of software development with the Scrum and CMMI. IET Softw. 6 (5), 416–422. doi:10.1049/iet-sen.2011.0193.
Maier, A.M., Moutrie, J., Clarkson, J., 2012. Assessing organizational capabilities: reviewing and guiding the development of maturity grids. IEEE Trans. Eng. Manage. 59 (1), 138–159. doi:10.1109/TEM.2010.2077289.
March, J.G., 1991. Exploration and exploitation in organizational learning. Organ Sci. 2 (1), 71–87. doi:10.1287/orsc.2.1.71.
McDaniel Jr, R.R., 2007. Management strategies for complex adaptive systems: sensemaking, learning and improvisation. Perform. Improv. Q. 20 (2), 21–41. doi:10.1111/j.1937-8327.2007.tb00438.x.
McHugh, O., Conboy, K., Lang, M., 2012. Agile practices: the impact on trust in software project teams. IEEE Softw. 29 (3), 71–76. doi:10.1109/MS.2011.118.
Melo, C.O., Cruzes, D.S., Kon, F., Conradi, R., 2013. Interpretative case studies on agile team productivity and management. Inf. Softw. Technol 55 (2), 412–427. doi:10.1016/j.infsof.2012.09.004.
Middleton, P., Joyce, D., 2012. Lean software management: BBC worldwide case study. IEEE Trans. Eng. Manage. 59 (1), 20–32. doi:10.1109/TEM.2010.2081675.
Misra, S.C., Kumar, V., Kumar, U., 2009. Identifying some important success factors in adopting agile software development practices. J. Syst. Softw. 82 (11), 1869–1890. doi:10.1016/j.jss.2009.05.052.
Mitleton-Kelly, E., 2003. Ten principles of complexity & enabling infrastructures. In: Mitleton-Kelly, E. (Ed.), Complex Systems and Evolutionary Perspectives of Organisations: the application of complexity theory to organizations. Elsevier Science Ltd, Oxford, UK.
Moe, N.B., Dingsøyr, T., Dybå, T., 2009. Overcoming barriers to self-management in software teams. IEEE Softw. 99 (99), 20–26. doi:10.1109/MS.2009.114.
Moe, N.B., Dingsøyr, T., Dybå, T., 2010. A teamwork model for understanding an agile team: a case study of a Scrum project. Inf. Softw. Technol 52 (5), 480–491. doi:10.1016/j.infsof.2009.11.004.
Nawrocki, J., Walter, B., Wojciechowski, A., 2001. Toward maturity model for extreme programming. Proceedings of the 27th Euromicro Conference 2001. September 04–06. Warsaw, pp. 233–239. doi:10.1109/EURMIC.2001.952459.
O'Reilly III, C.A., Tushman, M.L., 2008. Ambidexterity as a dynamic capability: resolving the innovators' dilemma. Research in Organizational Behavior 28, 185–206. doi:10.1016/j.riob.2008.06.002.
Ozcan-Top, O., 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.
Packlick, J. 2007. The agility maturity map – a goal oriented approach to agile improvement. Agile Conference 2007. 13–17 August, pp. 266-271. doi 10.1109/AGILE.2007.55.
Patel, C., Ramachandran, M., 2009. Agile maturity model (AMM): a software process improvement framework for agile software development practices. Int. J. Softw. Eng. 2 (1), 3–28 Available at http://goo.gl/FGe0eE.
Paulk, M., 2001. Extreme programming from a CMM perspective. IEEE Softw. 18 (6), 19–26. doi:10.1109/52.965798.
Perrow, C., 1981. Normal accident at three mile island. Society 18 (5), 17–26. doi:10.1007/BF02701322.
Pomeroy-Huff, M., Mullaney, J., Cannon, R., Sebern, M., Humphrey, W. 2005. The Personal Software Process (PSP) Body of Knowledge, Version 1.0, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, Pennsylvania, Special Report CMU/SEI-2005-SR-003. Available at http://goo.gl/fCBzUi.
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, 179, pp. 277–284. doi:10.1007/978-3-319-06862-6_21.
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. 2013–2217. doi:10.1007/978-3-319-06862-6_14.
Qumer, A., Henderson-Sellers, B., 2008. A framework to support the evaluation, adoption and improvement of agile methods in practice. J. Syst. Softw. 81 (11), 1899–1919. doi:10.1016/j.jss.2007.12.806.
Raisch, S., Birkinshaw, J., 2008. Organizational ambidexterity: antecedents, outcomes and moderators. J. Manage. 34 (3), 375–409. doi:10.1177/0149206308316058.
Ramesh, B., Mohan, K., Cao, L., 2012. Ambidexterity in agile distributed development: an empirical investigation. Inf. Syst. Res. 23 (2), 323–339. doi:10.1287/isre.1110.0351.
Schweigert, T., Nevalainen, R., Vohwinkel, D., Korsaa, M., Biro, M., 2012. 289-294. In: Mas, A. (Ed.), Agile maturity model: oxymoron or the next level of understanding. SPICE doi:10.1007/978-3-642-30439-2_34.
Sheffield, J., Lemétayer, J., 2013. Factor associated with the software development agility of successful projects. Int. J. Proj. Manag. 31 (3), 459–472. doi:10.1016/j.ijproman.2012.09.011.
Sidky, A., Arthur, J., Bohner, S., 2007. A disciplined approach to adopting agile practices: the agile adoption framework. Innov. Syst. Softw. Eng. 3 (3), 203–216. doi:10.1007/s11334-007-0026-z.
Sjøberg, D.I.K., Dybå, T., Anda, B.C.D., Hannay, J.E., 2008. Building theories in software engineering. In: Shull, F. (Ed.). Guide to Advanced Empirical Software Engineering, pp. 312–336. doi:10.1007/978-1-84800-044-5_12.
Snowden, D.J., Boone, M.E., 2007. A leader's framework for decision-making. Harvard Business Review. 85 (November (11)), 68–76 Available athttp://goo.gl/6NpWe.
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.
Spoelstra, W., Iacob, M., Van Sinderen, M., 2011. Software reuse in agile development organizations – a conceptual management tool. SAC '11: Proceedings of the 2011 ACM Symposium on Applied Computing, pp. 315–322. doi:10.1145/1982185.1982255.
Stacey, R., 1996. Complexity and Creativity in Organizations. Berret-Koehler Publishers, San Francisco.
Stacey, R., Griffin, D., Shaw, P., 2000. Complexity and Management: Fad or Radical Challenge to Systems Thinking?. Routledge, London and New York.
Sutherland, J., Jakobsen, C.R., Johnson, K., 2007. Scrum and CMMI level 5: the magic potion for code warriors. Agile Conference 2007, 13–17 August. pp. 272-278. doi: 10.1109/AGILE.2007.52.
Tiwana, A., 2008. Do bridging ties complement strong ties? An empirical examination of alliance ambidexterity. Strateg. Manage. J. 29 (3), 251–272. doi:10.1002/smj.666.

Tsoukas, H., 2005. Complex Knowledge: Studies in Organizational Epistemology. Oxford University Press, New York.
Tsoukas, H., Hatch, M.J., 2005. Complex thinking, complex practice: the case for a narrative approach to organizational complexity. In: Tsoukas, H. (Ed.), Complex Knowledge: Studies in Organizational Epistemology. Oxford University Press, Oxford.
Tuan, N.N., Thang, H.Q., 2013. Combining maturity with agility – lessons learnt from a case study. Proceedings of the SoICT'13, December 05–06. Danang, Viet Nam, pp. 267–274. doi:10.1145/2542050.2542072.
Tuckman, B.W., 1965. Developmental sequence in small groups. Psychol. Bull. 63 (6), 384–399 http://dx.doi.org/10.1037/h0022100.
Turner, N., Swart, J., Maylor, H., 2013. Mechanisms for managing ambidexterity: a review and research agenda. Int. J. Manage. Rev. 15 (3), 317–332. doi:10.1111/j.1468-2370.2012.00343.x.
Vidgen, R., Wang, X., 2009. Coevolving systems and the organization of agile software development. Inf. Syst. Res. 20 (3), 355–376. doi:10.1287/isre.1090.0237.
Weick, K.E., Sutcliffe, K.M., Obstfeld, D., 2005. Organizing and the process of sensemaking. Organ Sci. 26 (14), 409–421. doi:10.1287/orsc.1050.0133.
Yin, A., Figueiredo, S., Silva, M.M. (2011) Scrum Maturity Model: validation for IT organizations' roadmap to develop software centered on the client role. ICSEA 2011, The Sixth International Conference on Software Engineering Advances, pp. 20–29. 23–29 October, Barcelona. Available at http://goo.gl/SklUZr.
Yin, R.K., 2005. Estudo de Caso: Planejamento e Métodos, 3ª ed Bookman, Porto Alegre.

Rafaela Mantovani Fontana has a Bachelor's degree in Computer Science and a Master's Degree in Systems and Production Engineering. She is a doctoral student at the Pontifical Catholic University of Paraná and a professor at the Federal University of Paraná. Her research interests include agile software development methods, project management, software process improvement, software quality and complexity theory applied to management.

Victor Meyer Jr. is Professor of Strategic Management at the Post Graduate Program in Business Administration, Pontifícia Universidade Católica do Paraná, Brazil. He received his Master and Doctorate degrees from the University of Houston, USA and conducted postdoctoral studies at the University of Michigan, USA. He has been a visiting professor at the School of Public Services, DePaul University, Chicago and the author of books and academic journal articles published in Brazil and abroad. He is a member of the Editorial Board of the *Universidade em Debate*, a Brazilian journal devoted to the main issues in the field at both national and international contexts. His main research focuses on strategic management in complex organizations, with special interest in strategic practices in higher education institutions and hospitals.

Sheila Reinehr has a Bachelor's Degree in Mechanical Engineering, a Master's Degree in Informatics and a Doctorate in Engineering. She is a professor at the Pontifical Catholic University of Paraná and an experienced researcher in the following areas of computer science: software engineering, software process improvement, software quality, project management, software product lines and metrics.

Andreia Malucelli has a Bachelor's Degree in Informatics, a Master's Degree in Electrical Engineering and a Doctorate in Electrical and Computing Engineering. She is a professor at the Pontifical Catholic University of Paraná and an experienced researcher in the following areas of computer science: software engineering, artificial intelligence, organizational learning, ontologies, multiagent systems and healthcare information systems.