Maturity in Agile Software Development

Rafaela Mantovani Fontana[1],
Sheila Reinehr (supervisor)[2], **Andreia Malucelli (co-supervisor)**[2]

[1]Setor de Educação Profissional e Tecnológica
Universidade Federal do Paraná, UFPR
R. Dr. Alcides Vieira Arcoverde, 1225 – 81520-260 – Curitiba – PR

[2]Programa de Pós Graduação em Informática
Pontifícia Universidade Católica do Paraná, PUCPR
R. Imaculada Conceição, 1155 – 80215-901 – Curitiba – PR

rafaela.fontana@ufpr.br, sheila.reinehr@pucpr.br, malu@ppgia.pucpr.br

Abstract. *This paper summarizes the main objectives, research approach, and contributions of the doctoral dissertation named as "Maturity in Agile Software Development". It was developed in the Graduate Program in Informatics of the Pontifical Catholic University of Paraná, from 2013 to 2015. The study was developed with the objective to characterize maturity in agile software development, as a means to guide agile teams in software process improvements. Our results contribute to the software engineering body of knowledge by showing a picture of maturity in agile teams that goes beyond defining and controlling processes, as usually stated by reference models.*

1. Introduction

In the software engineering field, maturity models are used on a regular basis to guide improvements in software development processes. Each maturity model is founded on an underlying concept for maturity and the roadmap that the element (person, object or social system) follows to mature [Kohlegger et al. 2009]. In the current established maturity models for software development teams, maturity is achieved when continuous improvement takes place. To accomplish that, teams must define work processes, as well as standardize and quantitatively manage them [CMMI Product Team 2010].

As soon as agile software development methods [Beck et al. 2001] spread worldwide, researchers and practitioners started to implement both agile and CMMI-DEV simultaneously, to have the best of both worlds: agility to deliver to customers, with disciplined processes. Studies in the field have been reporting that CMMI-DEV, for example, should be used to help organizations institutionalize agile methods. They also point out that agile thinking guarantees that processes are implemented efficiently while responding to changes, and CMMI-DEV guarantees that all relevant processes are considered with appropriate discipline [Paulk 2001, Sutherland et al. 2007, Lukasiewick and Miler 2012].

However, when agile teams implement a software process improvement model to improve the way they work, such as CMMI-DEV, the extensive definition and the control of the process hinders sustaining agility at the highest maturity levels [Paulk 2001, Lukasiewick and Miler 2012]. There is a need, then, for models that guide improvements for agile teams, without hindering their agility. Some of such models have been proposed in last years [Özcan-Top and Demirörs 2013, Leppänen 2013]. An issue remains, though, because all these models prescribe the practices and stages of adoption that teams should follow. The problem with this prescription is that the adoption of agile methods is usually too context-dependent, and there is evidence that agile teams struggle to follow prescribed practices [Fontana et al. 2014b].

We developed this research, thus, founded on the practical need to provide improvement guidelines for agile software development teams – explicitly considering their nature of not following prescribed practices. World industrial context shows that agile software development adoption is growing and that teams are maturing their practices. In Brazil, where interest in adopting agile methods is also increasing [Melo et al. 2013], the results of this study might help agile teams improving their software processes and, as a consequence, improving their results for the organization and for the Brazilian software industry.

From the academic point of view, we identified in literature nine different agile maturity models to guide improvements in agile teams. They have been proposed since the Agile Manifesto [Beck et al. 2001] has been stated, with different aims, different structures and even different underlying concepts of maturity. Some of them consider that mature agile teams are those productive; some consider that are those who sustain agility; and some state that are those who accomplish project performance. We identified, then, there is an academic need to understand the nature of maturing in agile teams. Our research questions were defined to address this need. They were: *"what is maturity in agile software development?"* and *"how do agile teams get mature?"*

The objective of this doctoral dissertation was, therefore, to *characterize maturity in agile software development*. Our purpose was to characterize this maturity because the nature of the element one wishes to aid in the maturing process must be considered in order to build useful guidelines for improvement [Kohlegger et al. 2009]. Thus, to accomplish that, we defined two specific objectives: 1) to define maturity in agile software development; and 2) to identify the mechanisms that teams apply to mature in agile software development.

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

2. Theoretical Foundation

As a basis to analysing results in this study, we considered that agile software development teams are complex adaptive systems. This approach for studying agile teams has already been applied in previous studies [Vidgen and Wang 2009] and considers that these systems 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].

Members of an agile software development team – as members of an organizational system – are part of both a legitimate network and a shadow network [Stacey 1996]. The legitimate network consists of the links between agents that are formally and intentionally established by management or by principles that are widely accepted. The shadow network consists of the links spontaneously and informally established. Here there are not just flows of information, energy and action, but also flows of emotion, friendship, trust and other qualities [Stacey 1996].

Because of the relevance of the effects of these shadows networks, 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. Instead, complex systems must be managed with processes of sense-making [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. Patterns may then emerge if leaders conduct experiments that are safe to fail and decision making may thus be based on probing, sensing and responding [Snowden and Boone 2007].

For an adaptive system to survive and to prosper, it needs to balance processes of exploration and exploitation [March 1991]. 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]. March states that adaptive systems that engage in either one or the other might fail: too much exploration takes to too many undeveloped new ideas and too little distinctive competence; and too much exploitation takes to suboptimal stable equilibria.

If agile teams are complex adaptive systems that need a management strategy based on experimentation, and the evolution of these systems is a discontinuous process of exploration and exploitation, there is evidence that the path these systems take to maturity imply the development of an ability called organizational ambidexterity. Ambidextrous organizations are aligned and efficient in the management of current demands while simultaneously adaptive to change in the environment. Ambidexterity is identified by the behavioral capacity of a business unit to demonstrate both alignment and adaptation simultaneously, and has already been related to higher levels of performance [Gibson and Birkinshaw 2004].

Based on this theoretical foundation, which considers that organizations and, thus, agile software development teams, are complex adaptive systems, we proposed our Conceptual Framework (Figure 1). For the objectives of this study, we understand that agile teams 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 the theoretical basis we used for data analysis:

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

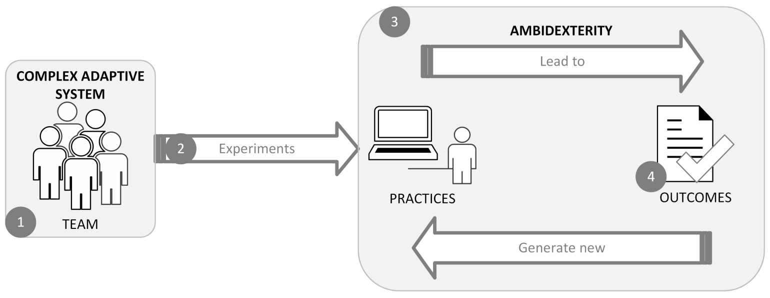

Figure 1. Conceptual Framework

3. Research Approach

With the objective to characterize maturity in agile software development, our study was guided by two research questions: *"what is maturity in agile software development?"* and *"how do agile teams get mature?"*. The first question was answered by applying the survey method and the second answer was answered by applying a mixed-method approach, in two main stages, as shown in Figure 2. The last research stage was dedicated to evaluating the results. Next subsections explain each research step.

3.1. Stage 1 - What is maturity in agile software development?

This was the first stage of the research, in which we were searching for a definition for agile software development maturity. To accomplish that, we performed an on-line survey with 51 practitioners.

This survey comprised two parts. In the first part of the questionnaire, we listed 85 agile and non-agile practices and respondents had to evaluate the perceived maturity of each practice. Then, on a five-point Likert scale, they classified each practice as 1 "No maturity"), 2 ("Somewhat Mature", 3 ("Mature"), 4 ("Very Mature"), or 5 ("Very High Maturity"). In the second part of the questionnaire, they answered an open-ended question: "Based on your experience, what is agile software development maturity?".

To analyse the maturity classifications given by practitioners we used statistical cluster analysis and grouped practices according to their perceived maturity. It allowed us to identify groups of practices that were considered more mature than others. To complement this analysis, we applied the content analysis technique to identify the concepts or terms that occurred the most in the answers for the open-ended question. By combining these two analysis, we proposed a definition for maturity in agile software development. The result of this stage was published in [Fontana et al. 2014a].

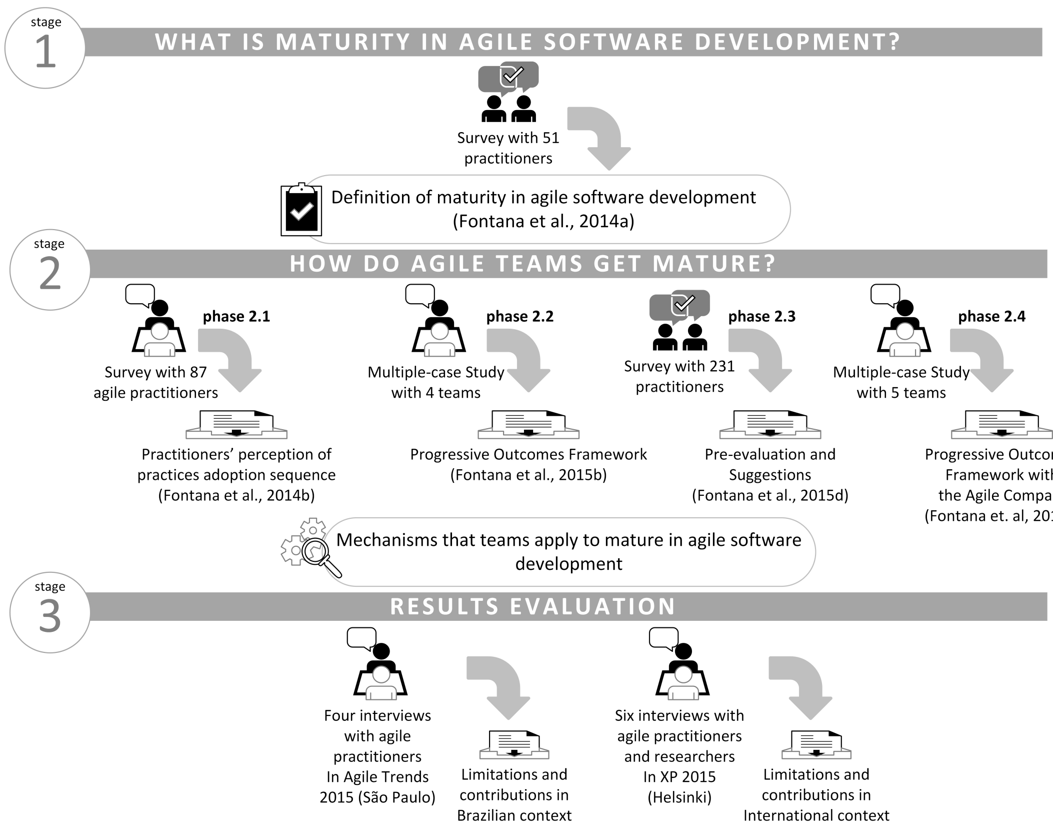
stage 1
WHAT IS MATURITY IN AGILE SOFTWARE DEVELOPMENT?
Survey with 51 practitioners
Definition of maturity in agile software development (Fontana et al., 2014a)
stage 2
HOW DO AGILE TEAMS GET MATURE?
phase 2.1
Survey with 87 agile practitioners
Practitioners' perception of practices adoption sequence (Fontana et al., 2014b)
phase 2.2
Multiple-case Study with 4 teams
Progressive Outcomes Framework (Fontana et al., 2015b)
phase 2.3
Survey with 231 practitioners
Pre-evaluation and Suggestions (Fontana et al., 2015d)
phase 2.4
Multiple-case Study with 5 teams
Progressive Outcomes Framework with the Agile Compass (Fontana et. al, 2015c)
Mechanisms that teams apply to mature in agile software development
stage 3
RESULTS EVALUATION
Four interviews with agile practitioners In Agile Trends 2015 (São Paulo)
Limitations and contributions in Brazilian context
Six interviews with agile practitioners and researchers In XP 2015 (Helsinki)
Limitations and contributions in International context

Figure 2. Research Approach

3.2. Stage 2 - How do agile teams get mature?

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

3.2.1. Phase 2.1

In this phase we conducted a preliminary survey on an agile conference to gain insight on how agile teams mature over time. In the questionnaire, we asked practitioners to assign a sequence to a list of practices, which should mean an ideal sequence for agile software development maturing. Besides it, we inquired respondents about the usefulness of a maturity model for agile software development.

We performed three hypothesis tests on collected data to: i) verify whether the practices were relevant to maturity in agile software development; ii) identify a trend in the numbering of the practices and place them in an essential, intermediate or desirable level of agile maturity; and iii) verify whether the opinions of inexperienced practitioners are different from those of experienced practitioners. We gathered 87 responses in this survey and published results in [Fontana et al. 2014b].

3.2.2. Phase 2.2

The objective here was to answer the research question on how agile teams get mature, but this time based on the observation of real cases, rather than on practitioners opinions. Based on the preliminary insights of Phase 2.1 and on our conceptual framework (Figure 1), we defined four propositions that should be verified in the study:

- The team plays a central role in agile software development maturity;
- Teams get mature in agile software development by combining exploration and exploitation activities, that is, through ambidexterity;
- The exact set of practices is not predefined at each maturity stage; and
- Teams evolve in agile development by departing from agile values, involved customer, planning and requirements; and, then, later invest in agile coding and agile testing.

We conducted, then, four case studies in this phase. For each case, we collected qualitative data, by interviewing team members and asking them to tell us a story of their agile software development adoption. We also collected quantitative data. Each team member provided, through a questionnaire, ambidexterity and project success perception. Qualitative data was analysed using content analysis technique and quantitative data was analysed using Chi-Square statistical tests. After each case was completed, a cross-case analysis was conducted, which resulted in the Progressive Outcomes Framework. This framework describes seven categories in which agile teams can accomplish certain outcomes in their maturing process [Fontana et al. 2015b].

3.2.3. Phase 2.3

In this phase we conducted a pre-evaluation of the Progressive Outcomes Framework. We performed lectures in 3 agile conferences presenting the Progressive Outcomes Framework and the underlying maturity concept. After the lectures, we asked attendees to answer a questionnaire. This questionnaire comprised seven statements that should be evaluated on a five-point Likert scale, ranging from completely disagree to completely agree. The statements were based on design-science validation guidelines [Hevner et al. 2004], which pose that *utility, quality and efficacy* of the artifact – in this case, the framework – should be evaluated. Results of this pre-evaluation were published in [Fontana et al. 2015d].

3.2.4. Phase 2.4

In this last phase we complemented the results obtained in Phase 2.2 with five more agile teams. We followed the same research protocol and, with evidences from more teams, we could extend the results of the Progressive Outcomes Framework. We summarized, for each outcome, how a team could identify whether it was accomplished or not. The result was a diagnosing tool for identifying maturity in agile software development, which we called "Agile Compass", published in [Fontana et al. 2015c]. The procedure to create this tool was the following:

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

3.3. Stage 3 - Results Evaluation

In this last stage, semi-structured interviews with agile practitioners and researchers were used to investigate in depth the utility, quality and efficacy of the research results. We also included an evaluation of the applicability of the findings in international context, given the limitation of the sample with Brazilian teams. For this purpose, we conducted ten interviews, four interviews with Brazilian agile consultants/practitioners and six interviews with agile researchers and practitioners from other countries, as shown in Figure 2.

4. Results

Following the research approach described in Section 3, we could characterize maturity in agile software development. We have accomplished that by answering our two research questions and by conceiving a diagnosis tool – The Agile Compass – as described in the next subsections.

4.1. What is maturity in agile software development?

The analysis of the data we collected at the first stage of this study has shown that maturity in agile software development means having an experienced team that:

- collaborates on projects by communicating and being committed;
- cares about customers and software quality;
- allows requirements to change;
- shares knowledge;
- manages source code and tests using tools, methods and metrics supported by infrastructure that is appropriate for agility;
- self-organizes at a sustainable pace;
- standardizes and continuously improves agile practices; and,
- generates perceived outcomes for customers and management.

To get to this definition, the responses of the 51 questionnaires were analysed using quantitative and qualitative data. As explained in Section 3.1, in the quantitative analysis we used the cluster analysis technique to group software development practices, considering the maturity classification to which they were assigned. As a result, we obtained twenty clusters of practices and named them according to the characteristic of the inner practices.

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

In the qualitative analysis – the content analysis of the open-ended question – we identified fifty-two key concepts relative to agile software development maturity, as pointed out by our respondents. By analysing the frequency of occurrence of the concepts and categories in the responses, we could identify that the most cited concepts in agile maturity definition are related to product quality and the use of methods and tools.

Results of the quantitative and qualitative analysis were combined to propose the definition presented in the beginning of this subsection and published in [Fontana et al. 2014a].

4.2. How do agile teams get mature?

To provide an answer to our second research question, we investigated nine agile teams, as described previously. This study comprised interviews, which collected qualitative data about the evolvement of agile practices; and questionnaires, which collected quantitative data about ambidexterity and perception of project success.

The cross-case analysis was driven by the cases study propositions. Our first proposition, which stated that *the team plays a central role in agile software development maturity*, was confirmed. We identified that team conduct is associated to maturity at the same time that we lack data that shows that definition and standardization of processes grow as agile practices evolve.

Our second proposition speculated that *teams get mature in agile software development by combining exploration and exploitation activities, that is, through ambidexterity*. This proposition could be confirmed with our quantitative data. Questionnaires responses showed that lower perception of project success was associated to lower perception of ambidexterity. Conversely, high perception of good performance was associated to a higher perception of ambidexterity. In addition to that, management appeared in the cases playing an essential role in the evolvement of the team [Fontana et al. 2015a]. Responsive teams can evolve to confident teams when they find space for autonomy. Ambidextrous management is the one that allows this autonomy by keeping team alignment with adaptability.

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

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

However, a pattern in the *outcomes* emerged from the comparison of the cases.

Our cross-case analysis uncovered seven categories of pursued outcomes - or evolvement threads - and how they evolved on actual agile teams. The categories are: practices learning, team conduct, deliveries pace, features disclosure, software product, customer relationship and organizational support. We represented them in the Progressive Outcomes framework for agile software development, as shown in Figure 3.

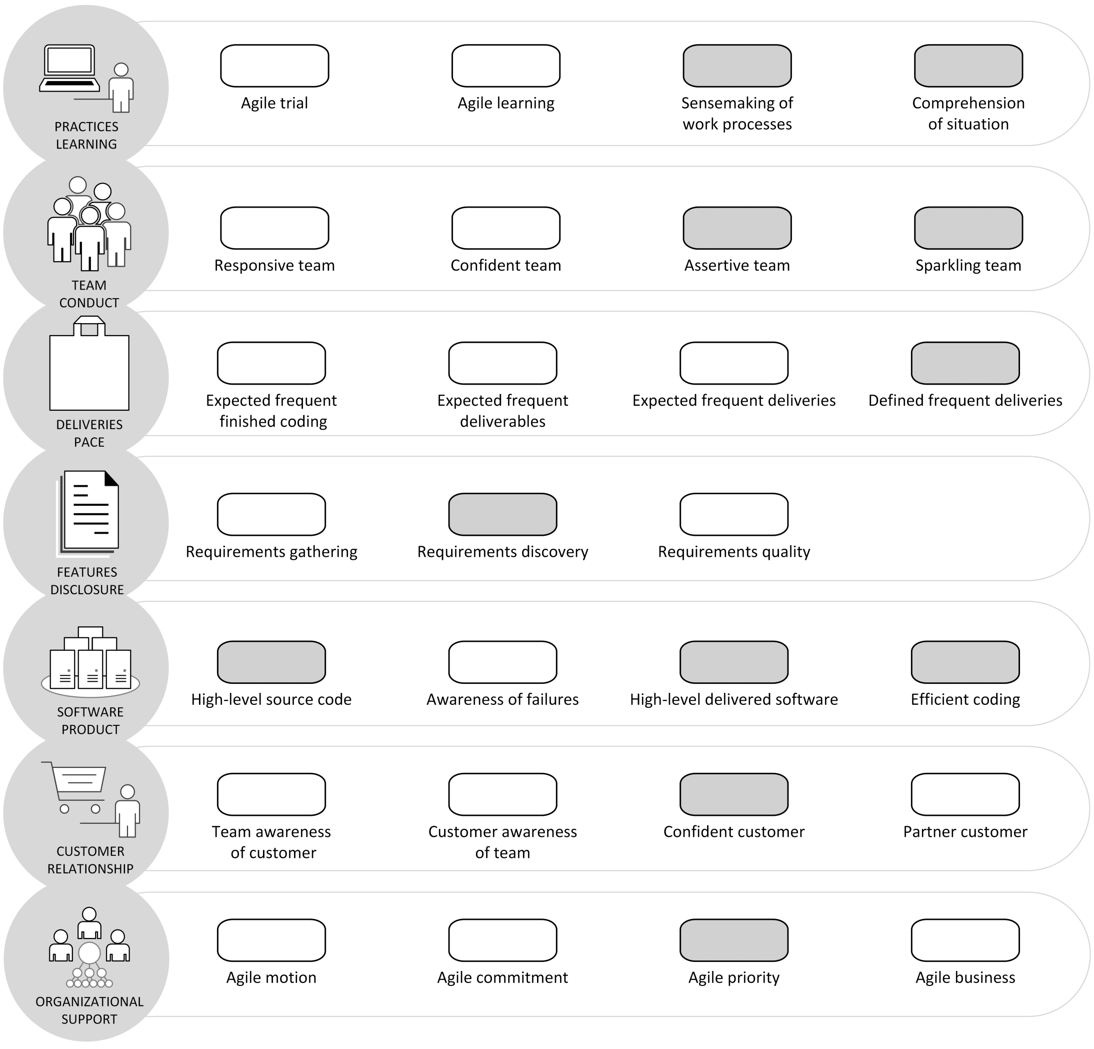

Figure 3. Progressive Outcomes Framework with mature outcomes highlighted

The practices learning category comprises the outcomes the teams pursue when they decide to change the way they work. The team conduct category describes how the team evolves in behavior with the use of agile methods. The deliveries pace category describes how the dynamics for delivering features changes with time. Features disclosure category describes outcomes teams pursue when defining the features the software will comprise. The outcomes for the software product category describe what the team pursues when it implements the practices to improve the software itself. The customer relationship category comprises the outcomes the team pursues when implements practices to improve its relationship with the customer. Lastly, the organizational support category describes the outcomes related to the position of the organization when providing support for agile transformation.

This framework represents a non-linear and dynamic, i.e. discontinuous, process of agile software development evolvement, as there are no stages or recommended sequence of evolvement. Besides, in a validation presentation that we made with the team leaders involved in the cases study, we presented which of the outcomes each team had accomplished. It became clear that each team evolves in the outcomes that are relevant to their business contexts. Mature outcomes could also be identified in the cases, and appear highlighted in Figure 3.

We complemented our main result - the Progressive Outcomes framework - with a diagnosis tool to allow the self-assessment of agile teams. We named it as the Agile Compass [Fontana et al. 2015c], as the objective was to provide a simple, slack and temporary picture of the outcomes a team has accomplished. To create it, we consolidated all the evidences across teams for each outcome. Based on these evidences, we created a checklist that teams may use to identify which outcomes have been accomplished and discuss the means to improve their processes.

Recent research has shown that the stage-based models for agile maturity have not been successfully validated and even criticized for not embracing the agility characteristics in the software process [Gandomani and Nafchi 2015, Gren et al. 2015]. We identified, with these results, that context dependence hinders the possibility of prescribing practices in agile maturity. Our theoretical foundation has shown that agile software development teams are complex adaptive systems. In this kind of systems, potential, creative, novel solutions emerge when the system is left to self-organize. That is, detailed codified routines hinder the emergence of optimal results. We have shown scientifically, thus, how real agile teams evolve and mapped this dynamics allowing teams to situate themselves and glimpse where new improvements should take place, with a clear view that there is a learning process and management strategies to be followed.

5. Discussion

Our definition, as well as the framework we proposed, places an emphasis on the importance of people. This central role people play in software development is reinforced by studies which show that increasing processes definition is hard to sustain and, as time goes by, people end up working with a minimum process [Coleman and O'Connor 2008]. There is plenty of evidence that, with agile methods, teams tailor their practices according to their contexts [Kirk and Tempero 2012, Bustard et al. 2013] and processes definition, therefore, is secondary.

For this reason, our results run counter initiatives that invest in maturing agile teams and organizations with the implementation of the CMMI-DEV reference model. We consider these initiatives valuable if there is no concern about sustaining agility and agile values such as self-organization, simplicity and adaptability. On the other hand, if teams wish to maintain agility, they cannot rely on these reference models. Currently published agile maturity models already address this issue by proposing other means to mature in agility [Leppänen 2013, Özcan-Top and Demirörs 2013].

What our study adds to the current body-of-knowledge in agile maturity is that context dependence hinders the possibility of prescribing practices. Our theoretical foundation has shown that agile software development teams are complex adaptive systems.

In this kind of systems, potential, creative, novel solutions emerge when the system is left to self-organize. That is, detailed codified routines hinder the emergence of optimal results. Our investigation of agile teams confirmed this lack of pattern in the adoption of the practices.

On the other hand, there was a pattern in the outcomes that teams pursued. When evolving to accomplish the outcomes, using whichever practices they feel like, teams also develop ambidextrous abilities. This ability reflects the capacity to be simultaneously aligned and flexible [Gibson and Birkinshaw 2004]. Even before Agile Manifesto, [Dybå2000] already identified that some of the key factors for successful software process improvement were "exploitation of existing knowledge, and exploration of new knowledge" [Dybå2000, p.259]. Our findings provided evidence that mature teams are also ambidextrous.

In accordance with the fact that there is not such a unique recipe for ambidexterity [Gibson and Birkinshaw 2004], our Progressive Outcomes framework considers this need to allow for ambidexterity development by defining the outcomes the team may pursue – the alignment – simultaneously making room for the team to implement the practices according to their contexts – the adaptability.

We observed, also, in a complementary study [Walter et al. 2015] that none of the outcomes described in our framework can be pursued and accomplished without effective management. Such management has the ability to keep a team aligned and at the same time flexible enough to reach outcomes. This ability is present in day-to-day decision-making and it seems to be strongly founded on concrete visibility of the current situation. This visibility allows the emergence of emotions that make people make sense of this situation. The use of simple metrics is essential to quickly understand the real situation and avoid making decisions based on assumptions. Simple metrics provide fast data to take simple actions that create visibility, and the cycle then restarts [Walter et al. 2015].

This is, actually, a continuous improvement cycle. Continuous improvement is also the aim of high-maturity teams which have adhered to CMMI-DEV. The difference we observed from the context of agile teams is that continuous improvement is not founded on extensive processes definition and measurement. They do not necessarily have to define work, then control it, then improve it, in this order. The continuous improvement cycle is quick (look, make sense and give simple responses) and performed since the very beginning of the work on the team [Clarke 2011]. Our study confirmed early conclusions from [Dybå2000] in the sense that, to accomplish improvements in software processes, commitment to learning is more important than to "best practice" models. We understand that this visibility-based continuous improvement leads to an ambidextrous behavior, which characterizes maturity in agile software development.

The dynamic of complex adaptive systems [Stacey 1996] may also be observed in the maturing process of agile teams. The outcomes a team accomplished during the maturing process may change as long as the environment changes: outcomes on a team recede if members are replaced, outcomes in a product recede if technology changes, outcomes in a customer recede if customer changes, outcomes in an organization recede with changes in business strategy.

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

6. Conclusions

This paper presented the objectives, research approach and results of the doctoral dissertation that aimed at characterizing maturity in agile software development.

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

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

Our results are threatened by surveys' samples sizes and this is the reason we conducted a mixed-method research approach. Findings are also limited to the contexts where data were collected. Firstly, to the Brazilian agile adoption context and, secondly, to the specific teams we have investigated. The Brazilian context of agile adoption is that of inexperienced practitioners. To mitigate this limitation, we have conducted validations with researchers from other countries. Although to some of them our results seemed familiar, we cannot assure applicability abroad. The limitation to the context of the teams we analyzed is a known limitation of case studies [Eisenhardt 1989], which deepen the analysis, but do not provide an extensive sample. We have worked to create a sample of companies with different profiles but, still, our findings are subject to replication in other contexts. Our findings are also limited to the evolution of agile practices within teams and, therefore, they do not address scaling of agile practices across team boundaries.

References

Beck et al. (2001). Agile manifesto. Available in http://agilemanifesto.org/. Accessed in

2013, May.

Bustard, D., Wilkie, G., and Greer, D. (2013). The maturation of agile software development principles and practice: Observations on successive industrial studies in 2010 and 2012. In *20th Annual IEEE International Conference and Workshops on the Engineering of Computer Based Systems (EBCS)*, pages 139–146.

Clarke, P.and O'Connor, R. V. (2011). An approach to evaluating software process adaptation. In *Proceedings of the 11th International Conference, SPICE 2011*, pages 28–41.

CMMI Product Team (2010). Cmmi for development, version 1.3 (cmu/sei-2010-tr-033). Technical report, Software Engineering Institute, Carnegie Mellon University.

Coleman, G. and O'Connor, R. (2008). Investigating software process in practice: A grounded theory perspective. *The Journal of Systems and Software*, 81(5):772–784.

Dybå, T. (2000). *Enabling Software Process Improvement: An Investigation of the Importance of Organizational Issues*. PhD thesis, Department of Computer and Information Science of the Norwegian University of Science and Technology.

Dybå, T., Maiden, N., and Glass, R. (2014). The reflective software engineer: Reflective practice. *IEEE Software*, 31(4):32–36.

Eisenhardt, K. (1989). Building theories from case study research. *Academy of Management Review*, 14(4):532–550.

Fontana, R. M., Fontana, I. M., Garbuio, P. A. R., Reinehr, S., and Malucelli, A. (2014a). Processes versus people: How should agile software development maturity be defined? *The Journal of Systems and Software*, 97:140–155.

Fontana, R. M., Meyer Jr., V., Reinehr, S., and Malucelli, A. (2015a). Management ambidexterity: A clue for maturing in agile software development. In *Lecture Notes in Business Information Processing*, pages 199–204.

Fontana, R. M., Meyer Jr., V., Reinehr, S., and Malucelli, A. (2015b). Progressive outcomes: A framework for maturing in agile software development. *The Journal of Systems and Software*, 102:88–108.

Fontana, R. M., Reinehr, S., and Malucelli, A. (2014b). Maturing in agile: What is it about? In *Proceedings of the 15th International Conference on Agile Software Development*, volume 6, pages 94–109.

Fontana, R. M., Reinehr, S., and Malucelli, A. (2015c). Agile compass: A tool for identifying maturity in agile software-development teams. *IEEE Software*, 32(6):20–23.

Fontana, R. M., Reinehr, S., and Malucelli, A. (2015d). Progressive outcomes: is it a handy approach to support agile methods process improvement? In *Proceedings of the XIV Simpsio Brasileiro de Qualidade de Software*, pages 17–21.

Gandomani, T. J. and Nafchi, M. Z. (2015). An empirically-developed framework for agile transition and adoption: A grounded theory approach. *The Journal of Systems and Software*, 107:204–219.

Gibson, C. and Birkinshaw, J. (2004). The antecedents, consequences, and mediating role of organizational ambidexterity. *Academy of Management Journal*, 47(2):209–226.

Gren, L., Torkar, R., and Feldt, R. (2015). The prospects of a quantitative measurement of agility: A validation study on an agile maturity model. *The Journal of Systems and Software*, 107:38–49.

Hevner, A. R., March, S. T., Park, J., and Ram, S. (2004). Design science in information systems research. *MIS Quarterly*, 28(1):75–105.

Kirk, D. and Tempero, E. (2012). A lightweight framework for describing software practices. *The Journal of Systems and Software*, 85(3):582–595.

Kohlegger, M., Maier, R., and Thalmann, S. (2009). Understanding maturity models results of a structured content analysis. In *Proceedings of the I-KNOW 09 and I-SEMANTICS 09*. Available at http://goo.gl/hnw7uh.

Leppänen, M. (2013). *Information Systems Development: Reflections, Challenges and New Directions*, chapter A Comparative Analysis of Agile Maturity Models, pages 329–343. Springer Science+Business Media.

Lukasiewick, K. and Miler, J. (2012). Improving agility and discipline of software development with the scrum and cmmi. *IET Software*, 6(5):416–422.

March, J. G. (1991). Exploration and exploitation in organizational learning. *Organization Science*, 2.

McDaniel Jr., R. R. (2007). Management strategies for complex adaptive systems. *Performance Improvement Quarterly*, 20(2):21–42.

Melo, C. O., Santos, V., Katayama, E., Corbucci, H., Prikladnicki, R., Goldman, A., and Kon, F. (2013). The evolution of agile software development in brazil. *Journal of the Brazilian Computer Society*, 19(4):523–552.

Özcan-Top, O. and Demirörs, O. (2013). Assessment of agile maturity models: A multiple case study. In *Software Process Improvement and Capability Determination, 13th International Conference, SPICE 2013*, pages 130–141.

Paulk, M. (2001). Extreme programming from a cmm perspective. *IEEE Software*, 18(6):19–26.

Snowden, D. J. and Boone, M. E. (2007). A leader's framework for decision-making. *Harvard Business Review*, page 6876.

Stacey, R. (1996). *Complexity and Creativity in Organizations*. Berret-Koehler Publishers.

Sutherland, J., Jakobsen, C. R., and Johnson, K. (2007). Scrum and cmmi level 5: The magic potion for code warriors. In *Proceedings of the Agile Conference 2007*, pages 272–278. DOI 10.1109/AGILE.2007.52.

Vidgen, R. and Wang, X. (2009). Coevolving systems and the organization of agile software development. *Information Systems Research*, 20(3):355376.

Walter, M., Tramontini, R., Fontana, R. M., Reinehr, S., and Malucelli, A. (2015). From sprints to lean flow: Management strategies for agile improvement. In *Lecture Notes in Business Information Processing*, volume 212, pages 310–318.

Weick, K. E., Sutcliffe, K. M., and Obstfeld, D. (2005). Organizing and the process of sensemaking. *Organization Science*, 26(14):409–421.