Agile Team Perceptions of Productivity Factors

Claudia Melo*, Daniela S. Cruzes†, Fabio Kon* and Reidar Conradi†
*Department of Computer Science, University of São Paulo, Brazil
†Department of Computer and Information Science, NTNU, Trondheim, Norway
({claudia,fabio.kon}@ime.usp.br) ({dcruzes,conradi}@idi.ntnu.no)

***Abstract*—In this paper, we investigate agile team perceptions of factors impacting their productivity. Within this overall goal, we also investigate which productivity concept was adopted by the agile teams studied. We here conducted two case studies in the industry and analyzed data from two projects that we followed for six months. From the perspective of agile team members, the three most perceived factors impacting on their productivity were appropriate team composition and allocation, external dependencies, and staff turnover. Teams also mentioned pair programming and collocation as agile practices that impact productivity. As a secondary finding, most team members did not share the same understanding of the concept of productivity. While some known factors still impact agile team productivity, new factors emerged from the interviews as potential productivity factors impacting agile teams.**

***Keywords*-productivity factors; agile methods; team productivity; empirical analysis**

I. Introduction

Low costs and time-to-market are the major drivers of software productivity improvements [1], [2]. Several studies have been carried out to address these issues by proposing and analyzing processes, methods, tools, and best practices [1], [3]–[6]. Despite efforts to increase software productivity, delivering on time and at budget are notoriously poor; and productivity has not shown much improvement over time [7].

Although productivity has been studied intensely [2], it remains a controversial issue. First, there are several concepts involved in its definition, such as effectiveness, efficiency, and performance, generating misunderstandings, and term overload. Second, the software productivity measurement is traditionally defined as the ratio of output (e.g., lines of code, function points, or implemented features) divided by input (e.g., time effort). In addition, software development is mental work involving knowledge creation or, at least, knowledge use as a dominant part of the work [8]. This concept may change the way we observe and interpret software productivity, since knowledge is complex and hard to evaluate.

Software productivity is also influenced by many factors, such as project characteristics, product complexity, or personnel skills [2], [9], [10]. Project managers are now more aware of the importance of factors that influence team productivity in projects [11]. To manage productivity effectively, it is important to identify the most relevant difficulties and develop strategies. Literature reviews [2], [9], [10] describe factors having significant impact on software productivity aiming to support companies with factor identification and selection.

Agile methods, such as Extreme Programming [12] and Scrum [13], have evolved as approaches that simplify the software development process. They aim to shorten the development time as well as deal with the inevitable changes resulting from market dynamics [14], [15]. Therefore, agile methods have been increasingly adopted and rapidly joined the mainstream of development approaches [16].

In a systematic review of empirical studies of agile software development [17], Dybå and Dingsoyr investigate what is currently known about the benefits and limitations of, and the strength of evidence supporting agile methods. They also identified four studies comparing the productivity of agile teams with the productivity of teams using traditional development methods; three of the four studies found that using XP results in increased productivity. Other studies have evaluated agile practices and productivity [18]–[21], but as companies are increasingly adopting agile methods, it is important to understand whether the factors as influencing productivity remain the same.

In parallel with our research, Hannay and Benestad [22] performed a study in productivity threats in a large agile development project, in which they uncovered ten productivity threats, as perceived by the project members. In particular, Petersen [23] recommends that new productivity factors have to be considered with the change in developing software, and old factors need to be re-evaluated when there is a new context. Our study aims to provide a better understanding of factors influencing the productivity of agile teams, as well as the concept of productivity they use. This research focus on the following research questions and sub-questions:

- *RQ1. How do agile teams define productivity?*
- *RQ2: What do agile team members identify as the main factors impacting on productivity?*
  - *How do these factors impact positively, or negatively on the productivity of agile teams?*
- *RQ3: Which agile practices are perceived to impact on a given team's productivity?*

For this purpose, we conducted two case studies in

the industry [24] to gain knowledge on the factors most perceived as impacting productivity according to the agile team members. Given the exploratory nature of our research, we decided to interview teams directly and ask their opinions, instead of using pre-existing productivity models (e.g. COCOMO [4]).

The rest of this paper is organized as follows. Section II gives an overview of software productivity factors found in the literature. Section III explains the concept of knowledge worker productivity. Section IV describes our research method and design. Section V presents the results and Section VI discusses the main findings and implications for research and practice. Section VII describes some limitations of this work. The last section concludes the paper and describes future work.

II. Productivity Factors

We identified three literature reviews [2], [9], [10] on productivity factors in software engineering. Table I presents the main factors identified from these reviews based on the most relevant sources and included only factors that had some empirical work studying the effect of the factor on productivity. We did not include studies that do not provide novel findings on productivity factors, such as work that use COCOMO factors without any new insight, or essays discussing productivity factors without any empirical evidence. Moreover, as we are considering team productivity, we will not discuss factors influencing individual or organizational productivity.

The factors in Table I include: product, personnel, project, and process factors. *Product* is related to a specific characterization of software, such as domain, requirements, architecture, code, documentation, interface, size, etc. *Personnel* factors involve team member capabilities, experience, and motivation. *Project* factors encompass management aspects, resource constraints, schedule, team communication, staff turnover, etc. *Process* factors include software methods, tools, customer participation, software lifecycle, and reuse. We highlighted factors that also appeared in our results.

III. Knowledge Worker Productivity

A Knowledge Worker (KW) has been described as a high-level employee who applies theoretical and analytical knowledge, acquired via formal education and experience, to develop new products or services [25]. Drucker states that knowledge-worker productivity is strongly related to the degree of autonomy, responsibility, and continuous lifelong learning a person experiences. He emphasizes that productivity is not only a matter of quantity of output, quality is at least as important [26].

Ramírez and Nembhard [8] summarized the most important dimensions that define the knowledge worker productivity. Table II describes the dimensions we consider more closely related to software development.

Table I
Team productivity factors from empirical studies

	Productivity factors	Description
PRODUCT	Reuse	Reuse of software products, processes, and artifacts, including components, frameworks, and software product lines.
	Software characteristics	System characteristics: architecture, complexity, domain, non-functional requirements, stability requirements, user interface, and software size.
PERSONNEL	**Team experience and capabilities**	Includes customer experience, domain knowledge and experience, generational experience, i.e., percentage of the development team already participating in two or more generations of software projects, programming language experience, staff capabilities and experience, and experience with tools.
	Motivation	Motivation to work on the project and in the company.
	Project Management	Includes aspects of quality of management, conflict management, task assignment, and administrative and formal coordination.
PROJECT	Resource constraints	Constraints such as timing, reliability, storage, team size, and project duration.
	Schedule	Concerns schedule compression and expansion.
	Team composition	Includes team size, team collocation, and staff turnover.
	Communication	Includes informal and face-to-face communication.
PROCESS	**Customer participation**	Refers to real customer involvement in project.
	Daily builds	Frequent integration of system components.
	Documentation	Use of artifacts to register project and product knowledge.
	Early prototyping	Early Prototyping stage involves prototyping efforts to resolve potential high-risk issues.
	Incremental and iterative development	Incremental approaches encompass various ways of producing a sequence of parts of a system, while iterative approaches involve a diversity of ways of producing parts of a system, trying them out, and feedback on user experience of production of new or revised parts.
	Modern programming practices	Use of top-down requirements analysis and design structured design notation, structured code, etc.
	Programming language abstraction	Level of abstraction of the language (e.g., Java is a high level language)
	Software methods	Methodologies and practices
	Tools usage	Use of CASE tools, IDEs, etc.

The Agile Manifesto relies on certain principles, in which the highest priority is to satisfy the customer through early and continuous delivery of valuable software [20]. The deliveries should be short-term and the delivered working software measures the project progress. Productivity in agile teams is therefore more related to the ability to manage deliveries and meeting deadlines, while maintaining quality and satisfying the customer. The main KW productivity dimensions related to the agile principles are *timeliness*,

Table II
KNOWLEDGE WORKER PRODUCTIVITY DIMENSIONS

Dimensions	Description (*adapted from [8]*)
1. Quantity	Accounts for outputs (quantities) and outcomes (quantification of qualitative variables such as customer and worker satisfaction)
2. Costs	Accounts for profitability, costs, etc.
3. Timeliness	Accounts for meeting datelines, overtime needed to complete the work, and other time related issues.
4. Autonomy	Accounts for independence and how many things a worker can do simultaneously.
5. Efficiency	Accounts for doing things right. Refers to any task, even if it is not important to the job. The task is completed meeting all the standards of time, quality, etc.
6. Quality	Accounts for how good the work is.
7. Effectiveness	Accounts for doing the right things. Refers just to the tasks that are important to the job, even if they are completed without meeting standards of time, quality, etc.
8. Project success	Accounts for overall result of work, considering decision-making, team interaction, communication, predictability, crisis management, documentation, transferability of work, etc.
9. Customer satisfaction	Accounts for the fact that the product needs to add value to the customer's business.

quality, and *customer satisfaction*.

IV. RESEARCH METHOD

We aim to investigate which productivity concept the teams use in their projects. We also want to identify which factors impact on agile team productivity and, specifically, how they impact on it. Finally, we want to explore whether agile practices impact productivity from the teams' viewpoint. To answer our research questions we performed two case studies in the industry [24], [27]. We chose to follow the teams for 6 months because the influence of some productivity factors may change across time, depending on the project context. For instance, the problem of staff turnover may be noticeable just after a member dismissal. Therefore, if we did not ask the interviewee frequently about specific factors that may have affected productivity, they may have forgotten the effect of the dismissals.

The criteria for case selection were: companies using agile (XP [12] and/or Scrum [13]) for at least 2 years; companies in different business segments, geographical location, size, structure, and culture; agile projects with, at least, four co-located developers; in progress for at least 6 months; and in different business areas. The unit of analysis is a set of two development projects, one company each.

Moreover, we drew up a non-disclosure agreement for companies to sign. This step was important to establish a formal link between researchers and companies, and ensure data confidentiality. The data collection was carried out in two Brazilian companies from Sept. 2010 to Feb. 2011.

A. Research context

Company 1 is a large financial corporation with more than 500 IT employees who had previously used plan-driven development processes. The company managers decided to adopt agile to increase team productivity. They have been using agile for two years. Project 1 is a re-development of an existing system for the financial market involving several institutions. The project started in March 2010, and is estimated to last for around two years. The team adopted several XP [12] and Scrum [13] practices and used 1-week iterations.

Company 2 has been delivering e-commerce and infrastructure services for over ten years, and has used only agile methods to develop software. It employs approximately 80 developers. Project 2 is a new development of an e-commerce system in a market with other competitors. The project also started in March 2010 but does not have a specific deadline, since they are developing software as a service, with continuous improvement and new functionalities. The project adopts several XP, Scrum, and Lean principles and practices.

Table IV describes the project profiles, considering guidelines provided by Kitchenham et al. [28]. Table III shows the level of adoption of agile practices by each project. If the team uses one practice fully, we assigned the term *full*. If they use just a few recommendations of the practice, we assigned the term *partial*. When they did not use it, we assigned the phrase *Do not use*.

B. Data collection and analysis

The main data collection methods were semi-structured interviews and informal face-to-face discussions with the team members. The data was collected throughout a period of 6 months, gathering opinions of different stages of the project.

The interviews were semi-structured (see the interview guide in Appendix A) to understand the factors impacting project productivity in the team's perception and how they impacted. One researcher conducted the interviews. Each interview lasted approximately one hour, and the interviewees were informed about the audio recording and its importance to the study. We interviewed 13 team members (see Table IV for more details), including developers, project managers and product owners, considering also different experience profiles.

We used thematic analysis to analyze the data, a technique for identifying, analyzing, and reporting standards (or themes) found in qualitative data [29]. According to Boyatzis [29], themes that are relevant to describe a phenomenon. To support the data analysis, we used a tool called NVivo (version 9) that allows the information classification into searchable codes.

After the interviews were transcribed, two researchers performed the data coding, naming all the possible factors

Table III
XP AND SCRUM PRACTICES ADOPTED BY THE PROJECTS

Practices	Project 1	Project 2
Code & Tests	Full	Full
Continuous integration	Full	Full
Daily deployment	Do not Use	Partial
Daily meeting	Full	Full
Energized work	Partial	Full
Incremental design	Full	Full
Pair programming	Full	Partial
Real customer involvement	Full	Full
Shared Code	Full	Full
Single code base	Full	Full
Sit together	Partial	Full
TDD	Partial	Full
Ten minute build	Full	Full
Negotiated scope contract	Do not Use	Partial
Planning game	Full	Partial
Retrospectives	Full	Full
Root cause analysis	Do not Use	Full
Slack	Full	Full
Stories	Full	Full
Team continuity	Partial	Partial
Weekly cycle	Full	Partial
Whole team	Partial	Partial

Table IV
PROJECT PROFILES

Characteristics	Project 1	Project 2
Team	6 full time developers, 2 part time developers, 1 scrum master, 1 project manager, 1 product owner (Total = 11 members)	4 full time developers, 2 part time developers, 1 project manager/coach, 1 product owner (Total = 8 members)
Language	Java	Ruby
Non-functional requirements	Reliability, Availability, Performance	Reliability, Availability
Reuse	High - Software product lines	High - Open source project
Stability requirements	High stability	Medium stability
Staff turnover	Considered medium by the project manager	Considered medium by the project manager
Interviewees	7 (3 full time developers, 1 part time developer, 1 product owner, 1 scrum master, 1 project manager)	6 (1 project manager/ coach, 1 product owner, 4 developers)

for productivity mentioned by the interviewees. Each code was discussed before including in NVivo.

To answer research question RQ1, we used the taxonomy of knowledge worker productivity proposed by Ramrez and Nembhard [8]. For RQ2 and RQ3, we chose a data-driven approach (Boyatzis [29] pp. 29-30), because we wanted the themes to emerge from the data. This was important to explore data without the influence of factors well known in the literature.

We then classified the patterns according to themes and, finally, all researchers interpreted and discussed the patterns. The data from the interviews were analyzed for each company and enabled the within-case analysis. Then, we aggregated the results to compare both companies in a cross-case analysis.

V. RESULTS

We describe below results from the interviews, focusing on answering the research questions posed in the introduction.

A. RQ1. How do agile teams define productivity?

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

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

B. RQ2: What do agile team members identify as the main factors impacting on productivity?

During the data analysis, we identified a total of 89 codes concerning factors impacting on productivity of the agile teams studied. The codes were organized into patterns originating from the main themes. We also ranked the themes by the number of sources, i.e., interviewees, who mentioned them. Only the most cited themes (mentioned by, at least, 30% of all interviewees) were selected to compose the final list of factors. Table V presents the most mentioned factors, a brief description, the percentage of citations in each project, the impact of the factor on their productivity (positive/negative), and the observed effects of the factor.

1) Team composition and allocation: Team composition is the configuration of member attributes in a team [30]. According to Bell [31], the team composition could be related to team performance because it affects the amount of knowledge and skills that team members need to apply to the team task.

As shown in Table V, *appropriate team composition and allocation* is considered a factor impacting on team productivity. The interviewees pointed out that small and mixed teams, with the required expertise, and being full-time allocated to the project are the most desirable attributes of team composition.

Team composition with different profiles and levels of knowledge was considered positive to team productivity, especially in Project 1. One possible explanation is that this team consists of business experts, experienced software architects, and beginners. Experienced team members contribute to the work adding knowledge, while the others

Table V
MOST PERCEIVED FACTORS, DEFINITIONS AND EFFECTS ON PRODUCTIVITY

Factor	Description	% / Number of responses	Impact	Effects
Team composition and allocation	• Various specialists in the team, even with different levels of knowledge (mixed team) • Full time team members • Everyone required to be involved with the system development is a part of the team, including the customer (whole team)	53% (7 out 13) (4 from Project 1, 3 from Project 2)	Positive/ Negative	• (+) Mixed teams o Experienced team members contribute to work adding knowledge, while others contribute being flexible • (+) Small teams o Better communication and alignment among team members o Easier conflict management and coordination o Sense of commitment and responsibility • (+) Full time team members o Team more focused, without work interruptions • (-) Whole teams o Important to have all required skills on the team, which is difficult to provide
External dependencies	• Need for external definitions or explanations • Need for waiting for another team to finish task, e.g. waiting for customer acceptance or for a component; interacting with external customers; publishing versions of system or of data model across different environments (integration, homologation, production)	46% (6 out 13) (3 from Project 1, 2 from Project 2)	Negative	• (-) Organizational definitions/policies do not consider agile team needs o Coordination of external dependencies is not compatible with the agile project pace • (-) External teams do not consider themselves part of team o This compromises teamwork and focus on project's overall goal
Staff turnover	• Exit or entry of people in the project team	30% (4 out 13) (2 from Project 1, 2 from Project 2)	Positive/ Negative	• (-) Loss of critical knowledge due to lack of documentation, even using job rotation • (+) Opportunity to learn new things and make improvements

contribute flexibility, as stated by a developer:

Some things contribute to productivity. We have very experienced people here, and less experienced ones that are more flexible (Developer, Project 1).

A lack of necessary expertise was also mentioned as a factor impacting on productivity. Both teams lacked personnel to play specific (and necessary) roles in the project, and consequently they faced problems completing the tasks. The major concern was about the lack of high-quality testing (for Project 1 and Project 2) and front-end design capabilities (for Project 2). Despite being important to have all the requisite skills on the team (the 'Whole team' agile practice), they are sometimes difficult to provide.

Very few people know certain things. Knowledge is very focused on some things... I think we are missing a test analyst to design test scenarios, because not everyone has the knowledge to construct them (Developer, Project 1)

To increase productivity, I would like to have a front-end staff on each teamThis profile is very hard to find (Project Manager, Project 2)

The interviewees mentioned that small teams lead to better communication and alignment. In addition, conflict management and coordination among team members are easier to deal with. Such responses confirms the findings from other studies [32]–[34]. That is, as team size increases, the number of necessary communication links between team members increases and, there will be more potential conflicts to manage.

In Project 2, some members left the project, and the remaining team members were allocated full-time to the project. A Developer said:

As we reduced the team, we are starting to focus more on what we want. Before, it was a bit messy and people did not perform certain tasks because they thought that other people would do it (Developer, Project 2)

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

After the reorganization, nobody needs to move to other activities outside the project. So, they are more focused. Everyone knows everything in the project (Product Owner, Project 2)

In Project 1, the team size increased over time and some team members noted it as a negative factor impacting on team productivity. From the team members' perspective, small teams also enable a better understanding of the big picture of the product. This is because fewer people need to learn and keep up to date on the product scope. In addition,

interviewees said that small teams help to increase the team's sense of responsibility and commitment.

2) External dependencies: There are several kinds of dependencies in a project. Shared resources, prerequisite constraints, simultaneity constraints, and the relationship of tasks and subtasks are common examples of dependencies among activities in a project [35]. *External dependencies* are also mentioned as a factor impacting on the team productivity (Table V).

Coordination processes are commonly used for managing dependencies among activities [35], [36]. Companies coordinate dependencies among teams or resources using some coordination strategies. For instance, when Project 1 delivers the system to the test environment, it has to submit some artifacts, such as data models, to the quality assurance (QA) team. The QA team is an example of resource shared among all enterprise projects of the company. It implements a coordination process first come/first serve [35] to manage requests from other teams. According to the team, this kind of coordination solution is misaligned with the pace of the agile project:

The other teams are not working at the pace of the project, they are not working in the (same) way... The organization is not ready yet (Developer, Project 1)

A similar problem occurs when Project 2 publishes the system to the corporate environment:

Currently, there is one factor that we are dealing with after a lot of feedback from our retrospective, which is about the relationship among the boundaries of development, testing, and production. Whenever we cross these boundaries, a bottleneck occurs. (Project Manager, Project 2)

Another external dependency problem occurs when an agile team decides to reuse components or existing systems, building a system of systems. External components and systems are often evolving and have their own lifecycle. Sometimes, agile teams need to wait for some components, which may compromise a timely delivery, the unsynchronized agile release pattern [37]. Figure 1 illustrate the lack of synchronization among teams that are working to the same project. The main team (Team A) requests components or services from external teams, such as Team B and C. While waiting for the requisitions to be completed, the main team thinks it is on track. However, when the planned system release date arrives, the team slips in integration problems. Project 1 reuses several components and has to implement interfaces to enable communication with other systems. The problem with the integration with components and systems is that:

The productivity at the end of the first phase of the project will be somewhat lower because we will have solved a lot of problems... which are the integration with other systems, publishing to a server with other business components. (Project Manager, Project 1)

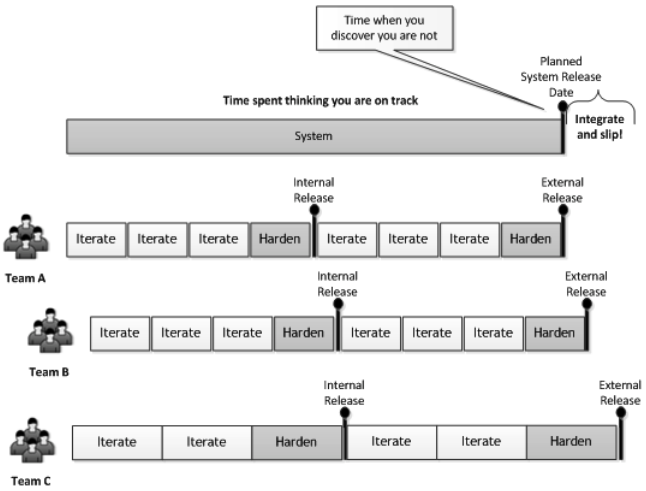

Figure 1. Unsynchronized agile release pattern (adapted from [37])

Therefore, these solutions to manage dependencies are not compatible with the needs of the team. For agile projects, it is not only important to manage dependencies, but also to resolve them in a timely manner. Thus, agile teams need to be more synchronized to achieve this final goal [37].

The interviewees also mentioned that the external teams sometimes are not committed to the project goal, but just with the execution of the requested task. In general, they do not consider themselves part of the team and tend to emphasize their importance in the process. A Developer complements:

There are a lot of roles, such as "I am the system administrator", "I am the QA" They don't understand that we're a multidisciplinary team. Reducing the conflict between these roles can simplify our work process (Developer, Project 2)

3) Staff turnover: Staff turnover is a common team risk for any software development project [38], [39]. Coram and Bohner [40] state that high turnover in a project can lead to loss of critical knowledge due to the lack of documentation. Even when teams use code reviews and job rotation the loss of a significant member of a team can still be catastrophic [40]. In our study, interviewees mentioned it as a factor impacting on team productivity (Table V).

There was some level of staff turnover in both teams studied. The teams perceived lower productivity due to staff turnover. In Project 1, at one specific time of the project, many team members left the project at once. Project 2 experienced lower, but frequent turnover. Both projects promote job rotation through pair programming.

There are things that impact the project negatively... One is the staff turnover. People who started the project are no longer here. Everybody now is new. (Developer, Project 1)

There was a drop in productivity at the end of the year when two developers left the project. (Project manager, Project 2)

Surprisingly, team members also mentioned a positive influence of staff turnover: the opportunity for the team to improve and grow. New team members can bring new ideas and experiences, leading the team to a more mature level. This not only relates to the team ability to deal with turbulent environments, but also to the continuous learning ability.

We see new people's arrival on the team in a different perspective. Maybe their proposals seem to be awkward for the team and they need to mature to a point at which maybe the proposed ideas will be good." (Developer from Project 2)

C. RQ3: Which agile practices are perceived to impact on a given team's productivity?

In order to answer RQ3, we coded all answers that relate an agile practice to team productivity. We then ranked the most cited practices and selected only those mentioned at least by 30% of interviewees. Table VI summarizes the two agile practices related to team productivity, a brief description of the practices, the percentage of responses used to rank them, the impact on team productivity, and the perceived effects of the factor.

1) Pair programming: Pair programming is the XP core practice and involves two programmers working collaboratively to develop software. Dybå et al. [19] conducted a meta-analysis on 18 studies regarding pair and solo programming. As a result, the authors provide guidelines for when to use pair programming based on the programmer expertise and the task complexity. They recommend that Intermediate and Senior developers should not work in pairs to solve easy tasks, because it could be counter-productive. Our results show that some team members also consider using pair programming to solve easy tasks a waste of time.

Despite being an advantage to use pair programming, when some activities are very simple, we do them individually. I mean, if you always work in pairs, I think there are times when you will lose productivity. (Developer, Project 1)

I think that pair programming brings a lot of benefit, works very well, but should not be used 100% of the time. Depending on the task, when it is very basic, repetitive, two guys working on that is unnecessary. (Developer, Project 2)

On the other hand, pair programming is also seen to contribute to increased satisfaction, keeping team members motivated [18], [41]. The results from our interviews were, however, different when interviewing intermediate and senior developers. Developers mentioned feeling demotivated when using pair programming all the time, because, sometimes, it is tiring or the task is simply too easy to be done in pairs. They also mentioned that, when solving complex tasks, they feel demotivated to work in pairs because they would like to have time to think about the problem alone before discussing it.

I think that pair programming is productive, but I don't like to work in pairs because I'm introspective. Sometimes, I like to be alone and start thinking, without much talking, solving the problem there, developing my solution. (Developer, Project 1)

Sometimes pair programming doesn't help. When you don't know in which direction you should go to solve complex problems. You want to use your own usual process to test things, trial and error, random exploration. This is not easy to explain to the others. (Developer, Project 1)

2) Collocation: The collocation of teams is an approach aiming to improve communication and collaboration among team members. Both Scrum and XP recommend collocation as an agile practice. When adopting this practice, companies also hope for productivity enhancement [42], but there are advantages and disadvantages with the use of collocation in software development [42]–[45]. Some projects reported significant productivity gains, and improvements in communication, spirit of teamwork, learning and motivation. Lack of privacy, work interruptions, lack of individual recognition, and some disconnection from the rest of the teams were mentioned as the negative side of collocation.

Collocation was mentioned by 30% of our interviewees. This result was stronger for Project 1. Team members in this project explained that collocated work helps to overcome the invisible existing barriers between teams in a hierarchical company. They noticed improvements in requirement negotiations and risk mitigations through the socializing atmosphere provided by the collocation.

The interviewees mentioned that their productivity vary in function of the workspace layout. Both projects work radically collocated [42] but they mention that is not enough to be in the same space. The layout (such as the desks positions and proximity) may also impact on their productivity. This factor was mentioned more by interviewees in Company 1 than in Company 2. This may be because Company 2 has invested in a customized layout that benefits working in pairs, while Company 1 has kept the same traditional workspace infrastructure.

VI. Discussion

Our results show that not only is productivity a diffuse concept but also, especially in the agile teams studied, it remains very much associated with *quantity*. In other words, for many of the interviewees, team productivity is the ratio of outputs by inputs. The story points delivered are the output and the iteration is the unit for the input, i.e., it represents the time spent to deliver the story points. Similarly, XP

Table VI
MOST PERCEIVED AGILE PRACTICES, DEFINITIONS AND EFFECTS ON PRODUCTIVITY

Agile practice	Description	% / Number of responses	Impact	Effects
Pair programming	Pair programming is the XP core practice and involves two programmers working collaboratively to develop software [18]	53% (7 out 13) (4 from Project 1, 3 from Project 2)	Positive/ Negative	• (+) Enables knowledge dissemination o Even when team members work on different schedules. o Especially when people have different backgrounds (e.g., domain vs technical). . • (+) Helps in creating commitment, supportiveness and trust. • (+) Increases the level of communication. • (-) Full time pair programming o Counter-productive and demotivating in easy tasks o Controversial issue when resolving complex tasks - it depends on the individual cognitive process of problem solving.
Collocation	Consists of placing team members near each other.	30% (4 out 13) (2 from Project 1, 2 from Project 2)	Positive	• (+) Helps in requirements negotiation and planning (re-planning) • (+) Improves communication and cooperation. • (+) Varies as a function of the workspace layout o Even when the team is collocated, team members want to be radically collocated

proposes project velocity as a productivity metric for team performance [12]. The drawback of this metric is that many teams use it as a way to show that the team is able to move rapidly, without considering that they may not necessarily be going in the right direction. The concepts of *timeliness*, *quality*, and *customer satisfaction* were not strongly associated with productivity by our interviewees, which contradicts the priorities of an agile team posed by the Agile Manifesto (see Section III).

Our results show that some traditional productivity factors (highlighted in Table I) are still impacting software development teams, even with the adoption of the agile practices. In addition, some specific factors emerged from the interviews as consequences of the use of agile practices in software development.

The factors impacting on productivity mentioned by our agile teams suggest that adopting agile methods should include several organizational actions in order to synchronize interdependent teams. In addition, these actions must address the integration of teams at an organizational level. Naturally, the organizational actions may vary with the type and severity of the team's external dependencies. Furthermore, collocation of teams seems not to be enough to reach high productivity in agile teams, but it seems that there is a need to invest in the workspace layout.

One striking result from our interviews was the connection of pair programming with tasks and motivation. When tasks are too easy or too complex, they may influence the motivation to work in pairs. As we have not found references to this phenomenon in the literature, we believe this is a point for further investigation in the use of pair programming for agile team productivity.

The following hypothesis can be derived from our results: i) the modality of the coordination process used to manage external dependencies impact on agile team productivity, due to the need for synchronization between the teams involved; ii) the degree of complexity of the task affects motivation to work in pairs, which in turn, affects the productivity; iii) productivity varies in function of the workspace layout, even when teams are radically collocated. There is a need for further investigation of these hypotheses in new empirical studies.

In comparison with Hannay and Benestad's study [46], there are some differences in the context and results. The context of their study consisted of 11 Scrum teams from three different subcontractors working on the same project, totaling up to 88 team members. Our two teams had 11 and 8 members in two different projects in two large companies. In terms of results, Hannay and Benestad found ten problem areas, and from these problems, only two were somehow related to our results: excessive dependencies within the system and difficulties when coordinating test and deployment with external parties. These two problems can be seen as part of the external dependency problem that was also mentioned often by our interviewees.

VII. LIMITATIONS

Although we have discussed the implication of our results for research and practice, the research reported here has limitations. First, our study was limited to members of two companies and two projects. The two projects had different characteristics, helping to overcome some issues with generalizability of our results. Within the projects, we were careful to interview members playing different roles on the team, so we could obtain different perspectives of productivity. As we pointed out in the discussion, the differences between our results and Hannay and Benestad's stress the need for further research to examine productivity factors in a wider variety of organizational and team settings.

Another limitation of our study is that we relied upon interviews to derive our results. The lack of standardization that it implies inevitably raises concerns about reliability. To reduce some of these concerns, we discussed and improved the interview protocol iteratively before data collection.

One researcher also visited the companies many times to better understand their context and observe the teams. The interviews were also conducted in different moments of the projects. For reliability of the analysis, two researchers coded all interviews individually and in a second step, they discussed each coding before including it in the software tool. One reliability issue that we did not overcome was the triangulation of the results with multiple methods of data collection.

VIII. Conclusion and Future Work

This study focused on investigating factors affecting productivity of agile teams. Based on our results, it is reasonable to conclude that there are some factors perceived more as impacting on the productivity of agile teams, namely: *Team composition and allocation*, *External dependencies*, and *Staff turnover*. All such factors are related to project management issues.

Regarding agile practices, *pair programming* and *team collocation* were the most cited practices impacting team productivity. As a secondary result, we found that the concepts of *timeliness*, *quality*, and *customer satisfaction* were not strongly associated with productivity by our interviewees, which contradicts the highest priority of an agile team in accordance with the Agile Manifesto [20].

A better understanding of the factors impacting on productivity can enable software teams to design interventions that allow software development teams to deliver software on time, with the desired quality and satisfying their customers.

The results of this study also indicate a number of directions for future research. We plan to complement the analysis performed in this paper with data from direct observation, team retrospectives, and burndown charts and by adding new companies and project data. Furthermore, we aim to design studies addressing the hypotheses that emerged during this study.

Acknowledgments

We are sincerely grateful to Dr. Tore Dybå for his valuable contribution to this work. We would like to thank the companies and their employees who contributed to this project. This research is supported by FAPESP, Brazil, proc. 2009/10338-3, CNPq, Brazil, proc. 76661/2010-2, and the Research Council of Norway, proc. 202657.

Appendix A

Interview guide

- What is your role in the project and how long have you been working on it?
- How is the team's way of work? What is your role in the team?
- How does the functionality prioritization work?
- How is the judgment done regarding the value that the team wants to deliver during prioritization?
- In you opinion, has the customer realized the delivered value in each iteration and release? Does he report this?
- In your opinion, has the team delivered value to the customer?
- What is the project status (scope, cost, time box, etc.)?
- What's your opinion regarding the project productivity? How do you realize this productivity?
- What's you opinion about project quality? How do you track the external quality? And internal? How do you handle the bugs?
- Is there any re-work level on the project?
- What do you think that most influences your team productivity?
- In your opinion, which changes were recently done in the team way of work that can have influenced any productivity variation?
- Do you consider the project motivating?
- Is there anything demotivating you in the project? And at the company?
- Is there anything in the agile methods that motivates you in anyway?
- Is there any kind of wasting which jeopardizes the project?
- If you could choose three things to increase the productivity, what would it be?
- Do you think that the agile methods usage increases the team productivity? Why? Is there something in agile that helps your own productivity or the productivity of your team?
- Is there anything in the agile methods that decreases your individual productivity or the team productivity? If so, what is it? Why?

References

[1] B. W. Boehm, "Improving software productivity," *Computer*, vol. 20, no. 9, pp. 43–57, 1987.

[2] A. Trendowicz and J. Münch, "Factors influencing software development productivity - state-of-the-art and industrial experiences," *Advances in Computers*, vol. 77, pp. 185–241, 2009.

[3] J. Vosburgh, B. Curtis, R. Wolverton, B. Albert, H. Malec, S. Hoben, and Y. Liu, "Productivity factors and programming environments," in *Proceedings of the 7th international conference on Software engineering*, ser. ICSE '84. Piscataway, NJ, USA: IEEE Press, 1984, pp. 143–152.

[4] B. W. Boehm, Clark, Horowitz, Brown, Reifer, Chulani, R. Madachy, and B. Steece, *Software Cost Estimation with Cocomo II*, 1st ed. Upper Saddle River, NJ, USA: Prentice Hall PTR, 2000.

[5] C. Jones, *Software assessments, benchmarks, and best practices*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2000.

[6] K. D. Maxwell, L. Van Wassenhove, and S. Dutta, "Software development productivity of european space, military, and industrial applications," *IEEE Trans. Softw. Eng.*, vol. 22, pp. 706–718, October 1996.

[7] C. Symons, "Software industry performance: What you measure is what you get," *IEEE Software*, vol. 27, no. 6, pp. 66 –72, Nov-Dec 2010.

[8] Y. W. Ramírez and D. A. Nembhard, "Measuring knowledge worker productivity: A taxonomy," *Journal of Intellectual Capital*, vol. 5, no. 4, pp. 602–628, 2004.

[9] M. R. S. Wagner, "A structured review of productivity factors in software development," Institut für Informatik - Technische Universität München, Tech. Rep. Technical Report TUM-I0832, 2008.

[10] S. C. de Barros Sampaio, E. A. Barros, G. S. de Aquino Junior, M. J. C. e Silva, and S. R. de Lemos Meira, "A review of productivity factors and strategies on software development," *Software Engineering Advances, International Conference on*, vol. 0, pp. 196–204, 2010.

[11] Z. Jiang, P. Naudé, and C. Comstock, "An investigation on the variation of software development productivity," *International Journal of Computer, Information, and Systems Sciences, and Engineering*, vol. 1, no. 2, pp. 72–81, 2007.

[12] K. Beck and C. Andres, *Extreme Programming Explained : Embrace Change (2nd Edition)*. Addison-Wesley Professional, November 2004.

[13] K. Schwaber and M. Beedle, *Agile Software Development with Scrum*, 1st ed. Upper Saddle River, NJ, USA: Prentice Hall PTR, 2001.

[14] D. Karlström and P. Runeson, "Integrating agile software development into stage-gate managed product development," *Empirical Softw. Engg.*, vol. 11, pp. 203–225, June 2006.

[15] M. Pikkarainen, J. Haikara, O. Salo, P. Abrahamsson, and J. Still, "The impact of agile practices on communication in software development," *Empirical Softw. Engg.*, vol. 13, pp. 303–337, June 2008.

[16] D. West and T. Grant, "Agile Development: Mainstream Adoption Has Changed Agility – Trends In Real-World Adoption Of Agile Methods," Forrester Research, Tech. Rep., Jan. 2010.

[17] T. Dybå and T. Dingsoyr, "Empirical studies of agile software development: A systematic review," *Information and Software Technology*, vol. 50, no. 9-10, pp. 833–859, 2008.

[18] V. Balijepally, R. Mahapatra, S. P. Nerur, and K. H. Price, "Are two heads better than one for software development? the productivity paradox of pair programming," *MIS Quarterly*, vol. 33, no. 1, pp. 91–118, 2009.

[19] T. Dybå, E. Arisholm, D. I. K. Sjøberg, J. E. Hannay, and F. Shull, "Are two heads better than one? on the effectiveness of pair programming," *IEEE Software*, vol. 24, pp. 12–15, November 2007.

[20] K. Beck, M. Beedle, A. van Bennekum, A. Cockburn, W. Cunningham, M. Fowler, J. Grenning, J. Highsmith, A. Hunt, R. Jeffries, J. Kern, B. Marick, R. C. Martin, S. Mellor, K. Schwaber, J. Sutherland, and D. Thomas, "Manifesto for agile software development," http://agilemanifesto.org/, 2001.

[21] F. Shull, G. Melnik, B. Turhan, L. Layman, M. Diep, and H. Erdogmus, "What do we know about test-driven development?" *IEEE Software*, vol. 27, pp. 16–19, November 2010.

[22] J. E. Hannay and H. C. Benestad, "Perceived productivity threats in large agile development projects," in *Proceedings of the 2010 ACM-IEEE International Symposium on Empirical Software Engineering and Measurement*, ser. ESEM '10. New York, NY, USA: ACM, 2010, pp. 15:1–15:10.

[23] K. Petersen, "Measuring and predicting software productivity: A systematic map and review," *Information and Software Technology*, vol. 53, no. 4, pp. 317 – 343, 2011, special section: Software Engineering track of the 24th Annual Symposium on Applied Computing - Software Engineering track of the 24th Annual Symposium on Applied Computing.

[24] J. Verner, J. Sampson, V. Tosic, N. Bakar, and B. Kitchenham, "Guidelines for industrially-based multiple case studies in software engineering," in *Research Challenges in Information Science, 2009. RCIS 2009. Third International Conference on*, april 2009, pp. 313 –324.

[25] P. F. Drucker, *Adventures of a Bystander*. New Brunswick, NJ: Transaction Publishers, 1994.

[26] ——, "Knowledge-Worker Productivity: The Biggest Challenge," *California Management Review*, vol. 41, no. 2, pp. 79–94, 1999.

[27] J. Gerring, *Case Study Research: Principles and Practices*. Cambridge University Press, 2006.

[28] B. Kitchenham, S. Pfleeger, L. Pickard, P. Jones, D. Hoaglin, K. El Emam, and J. Rosenberg, "Preliminary guidelines for empirical research in software engineering," *Software Engineering, IEEE Transactions on*, vol. 28, no. 8, pp. 721 – 734, aug 2002.

[29] R. E. Boyatzis, *Transforming qualitative information: thematic analysis and code development*. Sage Publications, 1998.

[30] J. M. Levine and R. L. Moreland, "Progress in small group research," *Annual Review of Psychology*, vol. 41, no. 1, pp. 585–634, 1990.

[31] S. T. Bell, "Deep-level composition variables as predictors of team performance: A meta-analysis," *Journal of Applied Psychology*, vol. 92, no. 3, pp. 595 – 615, 2007.

[32] J. D. Blackburn, G. D. Scudder, and L. N. Van Wassenhove, "Improving speed and productivity of software development: A global survey of software developers," *IEEE Transactions on Software Engineering*, vol. 22, pp. 875–885, December 1996.

[33] J. Blackburn, G. Scudder, and L. N. Van Wassenhove, "Concurrent software development," *Communications of ACM*, vol. 43, November 2000.

[34] F. P. Brooks, Jr., *The mythical man-month (anniversary ed.)*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1995.

[35] T. Malone, K. Crowston, J. Lee, and B. Pentland, "Tools for inventing organizations: toward a handbook of organizational processes," in *Enabling Technologies: Infrastructure for Collaborative Enterprises, 1993. Proceedings., Second Workshop on*, apr 1993, pp. 72 –82.

[36] T. W. Malone and K. Crowston, "The interdisciplinary study of coordination," *ACM Comput. Surv.*, vol. 26, pp. 87–119, March 1994.

[37] D. Leffingwell, *Scaling Software Agility: Best Practices for Large Enterprises (The Agile Software Development Series)*. Addison-Wesley Professional, 2007.

[38] T. Abdel-Hamid and S. E. Madnick, *Software project dynamics: an integrated approach*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1991.

[39] L. Wallace, M. Keil, and A. Rai, "Understanding software project risk: a cluster analysis," *Information & Management*, vol. 42, pp. 115–125, December 2004.

[40] M. Coram and S. Bohner, "The impact of agile methods on software project management," in *Proceedings of the 12th IEEE International Conference and Workshops on Engineering of Computer-Based Systems*. Washington, DC, USA: IEEE Computer Society, 2005, pp. 363–370.

[41] L. Williams, R. R. Kessler, W. Cunningham, and R. Jeffries, "Strengthening the case for pair programming," *IEEE Softw.*, vol. 17, pp. 19–25, July 2000.

[42] S. Teasley, L. Covi, M. S. Krishnan, and J. S. Olson, "How does radical collocation help a team succeed?" in *Proceedings of the 2000 ACM conference on Computer supported cooperative work*, ser. CSCW '00. New York, NY, USA: ACM, 2000, pp. 339–346.

[43] S. J. T. M. V. B. J. Eccles, M. and S. van der Watt, "The impact of collocation on the effectiveness of agile is development teams," *Communications of the IBIMA*, vol. 2010, 2010.

[44] P. J. Hinds and S. Kiesler, Eds., *Distributed Work*. Cambridge, MA, USA: MIT Press, 2002.

[45] F. Rafii, "How important is physical collocation to product development success?" *Business Horizons*, vol. 38, no. 1, pp. 78–84, 1995.

[46] J. E. Hannay, T. Dybå, E. Arisholm, and D. Sjøberg, "The effectiveness of pair programming: A meta-analysis," *Information & Software Technology*, vol. 51, no. 7, pp. 1110–1122, 2009.