Implementing Agile Practices: The Experience of TSol

Maria Cecilia Bastarrica
Computer Science Department
Universidad de Chile
celcia@dcc.uchile.cl

Germán Espinoza
TSol, Telecommunication Solutions
Santiago, Chile
g.espinoza.z@gmail.com

Jacqueline Marin
Computer Science Department
Universidad de Chile
jamarin@dcc.uchile.cl

ABSTRACT

Background: Implementing agile practices in software development processes promises to bring improvements in product quality and process productivity, but there are few reports of cases of failure to learn from. And more specifically, it is not always clear which practices work well in which contexts.

Aims: In this paper we present the experience of TSol, a small Chilean-based software company where some agile practices were added to an already formalized process. They intended to prove that this addition resulted in improved performance.

Method: We conducted a sequential explanatory strategy. First, an action research was applied implementing agile practices into the already existing process. Performance was measured in terms of the rate of rejected products in each process. Next, a survey was conducted by interviewing the development team members in order to know their opinion about the effectiveness of each of the applied agile practices. Finally, the obtained results were compared with scientific literature recommendations about agility implementation.

Results: There were clear improvements in performance. However, there was agreement about the usefulness of certain practices while others were not or they were even felt as a barrier for appropriate project development. Some of these results are consistent with the literature while others are not.

Conclusions: This work adds on the scarce lessons learned when agility implementation fails. Not counting on the few existing publications about failure cases of applying agility made TSol implement it in cases it was not recommended. Also, there are particular context circumstances that made TSol’s results different from past experiences.

CCS CONCEPTS
• Software and its engineering → Software development methods: Agile software development;

KEYWORDS
Software development processes; agile practices; empirical research

ACM Reference Format:

1 INTRODUCTION

The question of how software development should be organized has been debated by the software engineering community for decades.

The software development industry recognizes the need to define and manage processes in order to gain quality and productivity, as well as to apply practices that allow them to make the best use of the available resources [16]. Improving the ways in which software products are developed is a challenge that must be addressed carefully [25] due to software complexity [13].

Software development has adopted different forms during its evolution. It has followed highly formalized or structured approaches with a close management supervision, and also other more unstructured approaches. Many proposals for processes, models, methods, tools, techniques and concrete practices have been proposed to this end. However, evidence suggests that "there is no unique approach to software development" [6].

Traditional processes follow methods based on the quality of the artifacts, on the predictability of their processes and on architectural designs that can adapt to change before it has an adverse impact on the system [4]. Waterfall model established an era in which software development is recognized as a complex activity, centered on people, which has many difficulties if it is not properly organized [8].

Difficulties such as long development time, involvement of the client only at the beginning of the project, costly changes, difficulties for innovation and excessive documentation of these rigorous processes caused the advent of agile approaches. They aim to simplify the software process by avoiding bureaucracy and advocate for short time cycles, close involvement of the client and an adaptive rather than predictive strategy [1]. Many researches defend that agile methods provide a better way to address people’s needs, accelerate software development and improve quality and customer satisfaction. These and other reasons make industry to try to adopt the agile philosophy. However, agility is rarely used in its pure form [37] and can fail when applied in the wrong context [21].

The history of software development shows that the choice of a process is not a simple deterministic exercise, but depends on the situational characteristics of individual development environments [8]. Therefore, it is necessary to consider the application domain, as well as organizational, project and team characteristics, among others [42]. For example, highly critical systems such as aerospace or health control devices require exhaustive documentation, while small innovative software will benefit from agility.
As agile methods try to address the shortcomings of traditional software processes, their proposal stresses the limitations of the latter. In agile methods, software development is considered as the main character: better products result from higher qualifications in the development team, as well as their ability to collaborate among themselves and with the clients.

A whole series of successful experiences have been published, but only a few papers warn about situations where agility is not efficient or even not applicable at all [38]. Moreover, there is even less research published about the empirical evaluation of the effectiveness of the application of agile methods where drawbacks or failure cases are reported [2, 7, 14, 29].

In this paper we report the experience of the Improvement Support division of TSol, a small Chilean company, first applying a formalized process and then adding agile practices in a way of trying to improve efficiency and quality, as recommended in several technical and non-technical publications. To this end we conducted a sequential explanatory strategy. First, an action research was conducted where the original process and the one including agile practices were applied and measured in terms of products rejected by the clients. Then, development team members were surveyed to know their opinion about applicability and usefulness of each agile practice included in the process after applying the new process for three semesters. Finally, obtained results were compared with reported situations where agility is recommended or not.

As a result, the obtained process performance improvement was apparent. However, the analysis of the survey revealed that not all added practices were perceived as equally useful for project development, i.e., that the improvement was due to only some of the implemented practices. Moreover, some practices could not be implemented at all because of cultural or organizational barriers. For example, pair-programming was defined as part of the process, but management did not allow it to be applied as it was perceived as a source of inefficiency. Also, the Team responsible derived to be an actual project manager as all team members were used to work in that fashion. On the other hand, daily stand-up meetings were not only perceived as efficient but also emotionally fulfilling.

When comparing the obtained results with the scarce literature about failure cases of agility implementation, we found that some of them were consistent and knowing this issues would have been useful for TSol to avoid errors. However, some others are new as they depended on the specific characteristics of TSol and Chilean industry. We believe that these new context-dependent experiences can be useful for other companies when implementing agility.

The rest of the paper is structured as follows. Section 2 presents some background topics mainly about traditional and agile methodologies, as well as reported limitations to agility. The research design is presented in Sec. 3 describing the company, its original traditional process and the one including agile practices, as well as the methods for data collection and analysis. Results of the research are described in Sec. 4 and they are discussed in Sec. 5 comparing them with what scientific literature has reported about difficulties in applying agile practices. Finally, Sec. 6 presents conclusions and future work.

2 BACKGROUND

2.1 Traditional and Agile Processes

Since the beginning, software processes have been used to collect and organize knowledge about software development and, since then, a large number of approaches compete for the users’ favor [37]. According to several authors [4, 10, 31, 33, 34, 43], there are in practice two trends that guide software development, one following a rigid plan-driven approach and the other a set of slim agile methods. Traditional processes aim to address the entire life cycle of the software project while agile methods or more precisely agile software development methods or processes [20], intend to simplify the software process to the minimum to avoid bureaucracy. Each of the approaches, agile or traditional, emerged at a critical moment in the history of software development with well-defined purposes and contexts.

2.2 Application Contexts for Agile Processes

According to results of the study Status Quo Agile 2016/17 [19], in which more than 1000 people participated with a representation of more than 30 countries (mostly European), 85% of the users of agile methods surveyed agreed that the most widely used method is Scrum, followed by Kanban, Lean and DevOps [15]. More than 50% of the participants started using Scrum in 2014 or later, and 70% declare that they started using Kanban in the last 3 years.

The 11th Annual State of Agile Report [40] with representation from all continents (mostly North America), confirms that the adoption of agility continues to grow. The 94% of the respondents claimed that their organizations practice agile, although they also say that more than half of their teams still do not practice agile. Scrum and Scrum/XP Hybrid are the agile methodologies most commonly used by organizations, while the use of XP as an independent methodology decreases, even though the practices associated with XP are still frequent. The report states that 71% of the surveyed organizations have current or planned DevOps initiatives.

Both surveys agree that the main reasons to work with agile methods are: shorten time to market, improve the quality, reduce project risk, better change management, higher team productivity and better visibility of the project.

The results of [19] conclude that the success rate of agile methods is higher than that of traditional project management. According to the data of [19, 40] the improvements due to the implementation of agile methods are greater than the effort involved in the implementation itself. Even so, there are challenges to agile scaling such as the disagreement of the organizational culture with agile values and the lack of skills or experience with the methods [40].

In [19], it is also concluded that there is a trend to apply hybrid approaches more than purely agile. Different aspects such as the benefits brought by the agile philosophy and its difficulties in domains where traditional approaches still work fine, as well as the variable characteristics of different contexts, have led to defend the idea of combining both approaches [24, 26, 37, 44].

HELENA is an international study on the use of Hybrid dEveL-opmENt Approaches in software systems development. As part of
this study, a survey was conducted aiming to know which development approaches are used in practice and how they are combined, how such combinations were developed over time, and if and how standards impact the design and implementation of agile methods in practice. The first stage of HELENA survey [24] obtained answers from around 15 countries, mostly from Europe and North America. HELENA’s stage 2 [23] obtained answers from 31 countries around the whole world, and Chile among them. Results of these surveys show that companies tend to implement a balanced software approach that includes both methods traditional and agile, yet with a strong tendency toward agile.

Research such as [8, 21, 24], establish a series of factors that characterize the software development context and that may significantly affect the adoption of a software process at the level of the organization, project or development team. Some of these factors that should be taken into account are: governance type, business domain, maturity of the organization, level of innovation, culture, size of the system, personnel, team distribution, architectural effort, rate of change, and criticality, among others.

3 RESEARCH DESIGN

We next describe the company and its particular characteristics and then we detail the research protocol followed in each step.

3.1 The Company

Telecommunication Solutions (TSol) is an IT company that provides software development and support for several telecommunication companies in most Latin American countries. The operation center of TSol is located in Santiago, Chile. The company’s mission is to provide efficient IT services, identifying and taking advantage of the opportunities in order to contribute to its clients’ goals. TSol provides services related to all life cycle phases of the suite “Sistema Comercial Latinoamericano”² (SCL).

²Latin American Commercial System.

3.2 Research Protocol

In order to investigate whether the new process including agile practices improves process performance, we formulated the following research questions:

(1) Is the process including agile practices better in terms of the rate of products rejected?
(2) Which agile practices result useful and which ones do not?
(3) Are these results consistent with scientific literature?

For addressing this research we applied a mixed-methods approach, in particular a sequential explanatory strategy. We first...
conducted an action research and then a survey in order to aid in interpreting the results of the first stage.

3.2.1 Action Research. After applying the original process for four semesters, the new one including agile practices was defined in an attempt to improve software product quality and process efficiency. The motivation for using agile practices was to follow a trend in Chile. The new process was then applied for three semesters. Table 2 lists all the agile practices included in the new process. Also, new roles were defined such as Product Owner and Scrum Master, besides the development Team that was already part of the process. For example, Fig. 5 shows the Exploration phase once updated with some of the agile tasks defined. Notice that now, even though the process looks very similar, there are slight differences. For instance, there is an extra task Write user stories that is performed by the whole team and produces the set of user stories corresponding to the requirements being addressed. Also, there is another task, Obtain number of days that consists of estimating each of the user stories so that they can be prioritized afterwards.

The company has defined a quality metric for the area as the percentage of rejected products in each semester. A project is rejected when the client finds it has errors. In this case the rejection is
Table 2: Agile roles and practices in the new process

<table>
<thead>
<tr>
<th>Practice/Role</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simple design</td>
</tr>
<tr>
<td>Refactoring</td>
</tr>
<tr>
<td>Pair programming</td>
</tr>
<tr>
<td>Frequent releases</td>
</tr>
<tr>
<td>Test driven development</td>
</tr>
<tr>
<td>Standard code</td>
</tr>
<tr>
<td>Collective ownership</td>
</tr>
<tr>
<td>Continuous integration</td>
</tr>
<tr>
<td>Planning game</td>
</tr>
<tr>
<td>Sprint planning</td>
</tr>
<tr>
<td>Daily meeting</td>
</tr>
<tr>
<td>Retrospective</td>
</tr>
<tr>
<td>Product backlog</td>
</tr>
<tr>
<td>Sprint backlog</td>
</tr>
<tr>
<td>Scrum taskboard</td>
</tr>
<tr>
<td>Burndown chart</td>
</tr>
<tr>
<td>Scrum Master</td>
</tr>
<tr>
<td>Product Owner</td>
</tr>
<tr>
<td>Team</td>
</tr>
</tbody>
</table>

Table 3: Questions included as part of the survey

<table>
<thead>
<tr>
<th>Application of each agile practice</th>
<th>Yes</th>
<th>Adapted</th>
<th>No</th>
</tr>
</thead>
<tbody>
<tr>
<td>How</td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Result</td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Why</td>
<td>✓</td>
<td></td>
<td>✓</td>
</tr>
</tbody>
</table>

Table 4: Project performance with each process

<table>
<thead>
<tr>
<th>Semester</th>
<th>Rejected</th>
<th>Performance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial</td>
<td></td>
<td></td>
</tr>
<tr>
<td>First 2014</td>
<td>26.2%</td>
<td>Bad</td>
</tr>
<tr>
<td>Second 2014</td>
<td>40.0%</td>
<td>Bad</td>
</tr>
<tr>
<td>First 2015</td>
<td>40.7%</td>
<td>Bad</td>
</tr>
<tr>
<td>Second 2015</td>
<td>26.0%</td>
<td>Bad</td>
</tr>
<tr>
<td>Agile</td>
<td></td>
<td></td>
</tr>
<tr>
<td>First 2016</td>
<td>19.6%</td>
<td>Acceptable</td>
</tr>
<tr>
<td>Second 2016</td>
<td>18.0%</td>
<td>Acceptable</td>
</tr>
<tr>
<td>First 2017</td>
<td>8.3%</td>
<td>Good</td>
</tr>
</tbody>
</table>

accepted and measured as part of the metric. Rejected projects are addressed as new projects. It may also be the case that rejections are product extension; in these cases extensions are also addressed as new projects, but they are not considered for the metric. The company had defined thresholds for the quality metric: if rejection is lower than 15% operation is considered good, if it is between 15% and 20% it is considered acceptable, and bad results if rejection is higher than 20%.

The original process was applied for four semesters (between the first semester of 2014 and the second semester of 2015) and the percentage of rejected projects was recorded. Afterwards, the same procedure was applied during three other semesters (between the first semester of 2016 to the first semester of 2017) for the new process including agile practices. The performance of the process application was analyzed by comparing the evolution of these values, especially the difference between those semesters with the initial process and those with the process including agile practices.

3.2.2 Survey. A survey was designed in order to find out the opinion of all the members of the Improvement Support team about the applicability and usefulness of the agile practices prescribed by the new process. The survey was applied after the three semesters of applying the new process to all of the 20 people that form part of this department.

We inquired about the actual application of each of the agile practices included in the new process in the three semesters considered as well as their perception about their individual usefulness.

For each agile practice included in the new process, we asked if it was applied. Three answers were considered: Yes, when it was actually applied as prescribed, Adapted, when the practice needed to be modified in order to apply it, and No, when the practice was not applied for any reason. Agile practices that were not defined in the process were also considered as part of the survey even though we already knew that the answer would be No. In each case, the survey asked about details of How it was applied, the Results obtained and Why it was adapted or not applied. Table 3 summarizes the questions asked in each case.

4 RESULTS

Next we detail the results of applying each of the research steps.

4.1 Action Research Results

These results of this stage allows us to answer the first research question:

(1) Is the process including agile practices better in terms of the rate of products rejected?

The standard process was applied for four semesters in the Improvement Support area for solving a total of 137 issues. Then, the new process including agile practices was applied for three semesters for addressing 86 issues.

Table 4 reports the performance of the application of each process in terms of product rejection percentage. As can be seen, performance was consistently bad while applying the original software process, and it gradually improved after introducing agile practices. This gradual improvement is probably due to the slow adoption of the new practices. Therefore, the answer to question (1) is YES.

4.2 Survey Results

The survey was conducted in order to address the second research question.

Even though the new process included a whole series of agile practices and the team was encouraged to apply agile philosophy, some practices were actually applied and some others were not. Also, from those practices that were tried, only some of them were successful. Table 5 describes the list of agile practices and roles included in the process and the results of their application. We describe the information gathered during the survey with the development team after the three semesters when the new process
was applied. We considered practices both, successful and very suc-
cessful as applied, as well as those unsuccessful were considered as 
not applied. This has an implication about the questions answered 
as part of the survey, as stated in Tab. 3.

### 4.2.1 Successfully Applied Agile Practices

Table 5 describes the results of applying each of the agile practices included in the new 
process. We consider as “successfully applied” those practices 
that are described as either “successful” or “very successful”.

<table>
<thead>
<tr>
<th>Result</th>
<th>Practice</th>
</tr>
</thead>
<tbody>
<tr>
<td>Very Successful</td>
<td>Daily meeting</td>
</tr>
<tr>
<td></td>
<td>Retrospective</td>
</tr>
<tr>
<td></td>
<td>Product backlog</td>
</tr>
<tr>
<td></td>
<td>Sprint backlog</td>
</tr>
<tr>
<td></td>
<td>Simple design</td>
</tr>
<tr>
<td></td>
<td>Refactoring</td>
</tr>
<tr>
<td></td>
<td>Scrum taskboard</td>
</tr>
<tr>
<td></td>
<td>Standard code</td>
</tr>
<tr>
<td></td>
<td>Collective ownership</td>
</tr>
<tr>
<td>Successful</td>
<td>Pair programming</td>
</tr>
<tr>
<td></td>
<td>Test driven development</td>
</tr>
<tr>
<td></td>
<td>Planning game</td>
</tr>
<tr>
<td></td>
<td>Sprint planning</td>
</tr>
<tr>
<td>Adapted</td>
<td>Frequent releases</td>
</tr>
<tr>
<td></td>
<td>Continuous integration</td>
</tr>
<tr>
<td></td>
<td>Burndown chart</td>
</tr>
</tbody>
</table>

### 4.2.2 Adapted Practices

There are also some practices that were 
incorporated in turn: one developer builds and publishes a solution, and 
then, and after the developer completes the whole functionality and successfully 
exeucutes all unit tests, coding standards are applied. These include: 
trace recording, flag variables, log, norms, etc. This practice was 
slowly incorporated by developers and is currently indistinguish-
able from standard code. Refactoring was not addressed directly 
for improving legacy code.

**Scrum taskboard.** The Kanban board helped the team trace tasks 
that are being addressed in the sprint. First, a software-based tool was tried but a physical board resulted more appropriate and the 
practice was adopted.

**Standard code.** Architecture and coding standards were updated. 
The best results were obtained in Java-based applications such as 
EJB incorporated into Maven. In this way the ears could be 
incorporated in just minutes while it took a couple of days with 
the old process. Also, the newly generated ears are much lighter 
(kilos instead of megas) because now jars are not duplicated. Web 
applications are now lighter and simpler, and this has animated 
the team to look for other innovative solutions that could also be 
incorporated.

**Collective ownership.** Before deploying a solution, there is a meet-
ing with the Corrective Support team, provided that they would 
be in charge of maintaining it in the future, so that they are aware of 
the functional and technical features. A new SVN server was 
implemented for fostering changes, so developers in charge of in-
specting the code have their work easier and better organized. Once 
software units are versioned, they are taken out of the SVN. The 
only difficulty that was found is that sometimes developers forget 
to put into the SVN the source code they are working on.

### 4.2.2 Adapted Practices

There are also some practices that were 
incorporated in the new process. Managers in the company considered pair program-
ming as a loss in productivity due to “idle” people that may have 
been working on more useful tasks. Another strategy was imple-
mented in turn: one developer builds and publishes a solution, and 
another one inspects it, and is allowed to make all the changes that 
he or other people in the team may think appropriate. In this way 
there were always at least two people familiar with the code and 
the implemented functionality, so the backup situation provided 
by pair programming was achieved anyway. There were also fewer 
errors because code and ideas were shared among team members.

**Test driven development.** During 2016, DreamWorks was consid-
ered for performing automatic unit tests, mainly Java unit tests.
Provided that most software in the company was developed using Visual Basic 6.0, PL/SQL, pro*C, .Net, Lua, and Perl, no other framework was considered. Results were satisfactory in automating tests. However, the effort spent in programming tests was perceived as too much. Also, in the short term, and provided that the work corresponds to maintenance projects, these unit tests were not useful for regression tests. Therefore, automatic testing was shortly abandoned. However writing test before building the solution was adopted in a conceptual way: development used to be based on analysis and design documents, and in the new process, test planning was added to the requirements document so that it can guide development and then code certification.

Planning game. Product Owners send change requirements, a developer analyzes them and a solution is defined. To this end, the development team defines a user story, i.e., in this case user stories are not defined together with the user but just within the development team. These are not strictly user stories that are usually written in user terms and language, they are stated in commercial terms and acceptance criteria are defined in technical language. User stories are not handled to the Product Owner, instead they are used as documentation of the requirements and the solution to be implemented. The Product Owner is handled the project estimation in terms of days and a description of the solution mainly stating its scope. Sometimes the Product Owner also requires other specific documentation; in case building these documents is estimated as part of the time required for building the solution. Therefore, even though all stakeholders participate in planning, they do not follow a strict planning game.

Sprint planning. As soon as the Product Owner approves the proposed solution, the team plans the start and end dates of the sprint and the whole project, as well as the people that will take part in the project with their corresponding roles. This plan is refined in a team meeting and it is recorded and available in a Kanban board. The Product Owner defines the implementation priorities, but the amount of simultaneous development still requires negotiation.

4.2.3 Unsuccessful Practices. There are a series of agile practices defined as part of the new process. However, during its application, these practices could not be applied or were soon abandoned.

Frequent releases. The intention was to produce and release frequent product increments. However these releases needed to be received, certified and accepted by the client. This put stress on the resources the client should count on for this task. It required a great deal of negotiation to define the series of releases that the client was willing to receive. This number was always lower than desired, but since TSol develops solutions for the same client in each Latin American country, they have their own deployment plans and TSol did not have much power in this negotiation. This situation derived in having a unique release at the end of the project.

Continuous integration. This practice was initially implemented but it was soon abandoned. Very rarely there is more than one iteration provided that all development corresponds to Improvement Support. Continuous integration was perceived as unnecessary: each developed product was delivered to the Product Owner directly once completed. Continuous integration within iterations was not considered either.

Burndown chart. This practice was applied initially but it was soon abandoned because the difference between planned time and the actual time was always marginal, and therefore this chart did not add any useful information.

4.2.4 Agile Practices not Included in the Process. There were some agile practices that could have been included as part of the new process, but they were not. We also asked, as part of the survey why they were not included.

Client on site. For the company’s characteristics, signing a contract with clients does not imply that there will be expert users available for collaborating in constructing the solutions they require. Also, most of the clients are abroad and therefore it is difficult to count on users without incurring in a substantial increase in the budget. Nevertheless, when clients ask for modifications with fuzzy requirements, there is a contract agreeing that the project will continue, but it is clearly stated that it may be necessary to count on further negotiations in the eventual case that the scope needs to be changed or the deadlines adjusted. Even though TSol did not attempted to apply client on site, this issue brought as a consequence most of the adaptations that other practices needed.

40 hour week. In practice, no team member works less than 42 hours per week, even though the improvement Support area is trying to make this time more flexible for making work compatible with family time. Long working hours, and even working on weekends is a work-life balance that should be address across all software industry in Chile since it is a cultural issue that should be changed before this agile practice could be actually applied.

Metaphor. Metaphors are created in order to define a conceptual system abstraction that can be shared among all team members, including users and clients. However, and considering the company’s reality and that users and clients would not be accessible for validating the metaphor, it was decided not to use it. The shared knowledge a metaphor could have brought was supported with the knowledge about dealing always with the same software systems and also by daily meetings.

Delivery plan meeting. In this activity, the Product Owner sets priorities. In the case of TSol, plan approval and priorities were issued just through emails and personal conversations. There were no coordination problems, so the company kept this way of working.

Revision meeting. The team meets in order to review the product. Not as prescribed by Scrum, this meeting does not include the Product Owner because he is concerned about the product functionality and not necessarily its internal characteristics. In any case, when the team invited the Product Owner to the meeting, he was not interested. On the other hand, people from Corrective Support take part in the meeting before deployment considering that they would be in charge of maintenance.

4.2.5 About the Roles. Typical roles in agile development were defined for TSol with special characteristics as stated in Tab. 2. Not all of them resulted equally successful.
Scrum Master. This role was taken as an inspiration; instead a Team Responsible was defined. This role should be in charge of supervising the activities of all team members, as well as executing himself the same kinds of activities; he is also responsible for ensuring continuous process improvement. This last responsibility is typical for a Scrum Master, but the supervising activity is more likely to be performed by a Project Manager.

Product Owner. This role is in charge of defining and setting the priority of the requirements that the Improvement Support team will address. These are typical responsibilities of a Product Owner, however, in this case he was almost never collocated with the development teams and therefore he was not able to participate in most of the meetings and thus acted just as a client.

The Team. This is a typical team performing all software development related activities. Initially, they had little or no experience with agility and they learned as the projects progressed. They were very enthusiastic in participating and liked being considered as part of an horizontally organized team with little hierarchies.

Therefore, the answer to the second research question is summarized in Tab. 5:

(2) Which agile practices result useful and which ones do not?

4.3 Threats to validity

According to the recommendation of [18], we analyze three dimensions of threats to validity of our survey: content, criterion and construct validity.

Content validity. Content validity refers to the appropriateness of the survey instrument. We chose to include as part of the questionnaire all the agile practices that the company tried to implement, as well as those that are generally discussed in the literature as related with the context. In this way we think we are covering all the relevant results that could have been addressed.

Criterion validity. Criterion validity refers to as what extent the results could be generalized as predictors. Most successful/non-successful practices applied were consistent with reported application contexts. However, those that do not coincide could be due to particular Chilean software development context or other factors that were not taken into account. In this cases, more research is needed for confirming these results.

Construct validity. Construct validity represents the degree to which the questionnaire measures what it claims to be measuring. Even though there are slight differences in opinion about successfulness, all interviewed people agree in their results. This issue gives us confidence on our instrument.

5 DISCUSSION

This section intends to address the third research question:

(3) Are these results consistent with scientific literature?

Despite all the advantages that agile methods provide, the process of adopting the approach or transforming an organization toward agility is a challenge [21]. According to [33], changes not only affect the development process itself, but also they need to take into account a whole series of other factors: communication means, client participation, requirements, change management, management style, people and current processes.

In this section we discuss general difficulties about the application of agile approaches reported in the literature, and then compare them with the context of TSol. We try to understand if the difficulties in the adoption of certain agile practices in the company could have been prevented if the company had been aware of the available research in the area.

Table 6 from [27] summarizes several difficulties faced in the application of agile methods that have been reported in the literature. We now discuss each of these difficulties with respect to what happened when implementing agility in TSol.

Customer involvement in TSol. According to Boehm and Turner [4], the success of agile development depends on having customers representatives who are collaborative, representative, authorized, committed and knowledgeable (CRACK). There are also other related causes of failure such as changing the client on site without a timely transfer [39], or having several diverse clients [39], as well as bad communication between the client on site and other stakeholders [43]. There is consensus that client on site is one of the most stressful issues for the application of agile methods [4, 11, 31].

In the case of TSol, Client on site was one of the practices that was not even tried because of the characteristics of the company, and therefore it is not surprising that most practices that involved the client’s participation were not successful or needed adaptation, such as Frequent releases, Planning game or Sprint planning.

Professional demands. Agile methods are highly demanding when considering professional skills in order to be successful [4, 11, 22, 32, 33, 40]. Also, if team members do not count on comparable skills, development will not be effective [11]. Similarly, the lack of experience may generate big delays when new practices are implemented [33].

In TSol, teams were formed by an experienced project manager and a couple of less experienced developers. Therefore, when the agile process was implemented, product rejection rate slowly decreased; in fact the effect was only acceptable in the first two semesters, and good only in the third semester. Therefore, as long as the team remains the same, agility could succeed in the long term. However, and probably due to the difference in experience, they were never able to work completely as a horizontal team, and the manager remained as a manager even after implementing agility. Chilean software industry has a large shortage of skilled

<table>
<thead>
<tr>
<th>Table 6: Reported difficulties of agile methods</th>
</tr>
</thead>
<tbody>
<tr>
<td>Difficulty</td>
</tr>
<tr>
<td>Customer does not keep CRACK</td>
</tr>
<tr>
<td>Professional skill-specific demands</td>
</tr>
<tr>
<td>Agile development does not scale well</td>
</tr>
<tr>
<td>Lack of suitability for specific product domains</td>
</tr>
<tr>
<td>Architecture does not get enough focus</td>
</tr>
<tr>
<td>Some practices are discarded after a period of time</td>
</tr>
<tr>
<td>Increased testing effort</td>
</tr>
<tr>
<td>Risks in knowledge and project management</td>
</tr>
<tr>
<td>Increment of technical debt</td>
</tr>
</tbody>
</table>

...
software engineers and most of the workforce is formed by more or less experienced programmers. This situation cannot be directly addressed by a company by itself.

**Scalability.** Scalability can be considered from different points of view: number of people, product size, project time duration, product life cycle, distributed development, high risk or complexity. Scalability has been reported as a challenge for agile methods [4, 9, 11, 30]. In fact, it is a significant limitation and it is still an open issue to be addressed by researchers [33].

Having the clients abroad can be considered as a specific kind of distributed development as is the case of TSol, and this makes agility challenging. On the other hand, small development teams (3 to 4 people) working in small projects (support requirements) seems like a promising context for agility.

**Specific domains.** Several scientific works agree that agile methods have serious limitations in safety-critical domains (e.g., military or health care) and legacy systems [2, 4, 5, 24, 30, 33, 41] mainly because of simple design, refactoring and the lack of documentation.

Considering this issue, TSol is a good context for agility since they develop software that is commercially important but not safety-critical. However, projects mainly address maintenance evolution and therefore they involve legacy applications; in this case documentation could be a limitation but it did not happen probably because the application domain is well known for people in the Improvement Support area.

**Architecture focus and technical debt.** Another limitation of agile methods is the lack of attention received by design and architecture [3, 4, 21, 28, 30, 35, 36]. These issues bring as a consequence a grow in technical debt. According to Kruchten et al. [22], technical, quality and maintainability debts are not only caused by pressure on schedules, but also carelessness, lack of education, poor processes, non systematic verification of quality, or basic incompetence.

Even though agile practices do not focus on architecture and long term design, the *Simple design* practice was addressed from the architectural point of view. In this sense the approach was not strictly agile, but it was found highly successful among TSol developers.

**Discarded practices.** A study by Petersen [30] suggests that the lack of professional skills for agile development results in abandoning certain agile practices as time passes. The most frequently abandoned practices are: pair programming, test driven development, and continuous integration. Other works reporting similar results [11, 17] indicate that pair programming is perceived as an exhausting task whenever partners do no count on the same qualification.

Some of these reported cases happened in TSol. Pair programming was not strictly adopted, but due to managerial decisions. Test driven development was used just as an inspiration, and this was because team members were not knowledgeable in this practice, and probably neither was the Team Responsible. As the Product Owner was almost never near the Team, most activities that needed interaction, like planning meetings, could not be strictly applied.

**Testing effort.** A case study conducted by [30] concluded that in several small projects, independent tests are performed only partially or they are omitted completely. This makes that the burden of testing falls in the latest system version, demanding a huge effort [32, 33]. Also, a big effort is required for performing continuous testing because creating an integrated environment is difficult for different platforms and system dependencies [31].

Test driven development was seriously simplified in TSol, and was partially compensated with a second developer reviewing each piece of code. On the other hand continuous integrations was soon abandoned, and this could be one of the causes that contributed to the still remaining cases of product rejection.

**Knowledge and project management.** Boehm and Turner [4] highlight the risk of completely relying on the tacit knowledge assumed by agile methods, mainly in large teams provided that team members are not so interchangeable as needed [11]. This was not an issue in TSol provided that the team remained almost completely constant along the seven reported semesters. However, considering the high responsibility assigned to the Team Leader, similar to a Project Manager, it cannot be assured that the results would be the same if he were changed.

As an answer to the third research question, we can conclude that some of the results are consistent with what was published in the scientific literature but others are not.

## 6 CONCLUSIONS AND FUTURE WORK

In this paper we conducted a sequential explanatory study in TSol, a Chilean software development company. First, an action research was applied. Here, an original software process was rigorously defined and applied for four semesters, then some agile practices were added and applied for other three semesters. In both cases the context was the same: type of software developed, number and experience of the development team members, as well as the client. Performance was measured along the seven semesters in terms of percentage of rejected products. This metric improved considerably after introducing agile practices. Afterwards, and in order to try to understand why and how the new process brought this improvement, a survey was conducted asking involved developers about the application of each agile practice and their results. Finally, we compared these results with published scientific literature about applicability contexts for agility.

Even though clients were almost never around for interacting with the development team, practices that generally would have included the client such as Sprint planning or Frequent releases could be successfully adapted with almost no drawbacks. On the other hand, the fact that management would not allow Pair programming and the Team responsible soon became a project manager can be due to particular characteristics of Chilean society that is quite hierarchical and where managers hold much more power within organization than technical professionals. Their salaries are also much higher. The Chilean Ministry of Economics and the software industry association are aware of these issues.

We also concluded that most of the unsuccessful situations could have been avoided if people in charge of defining the process had known about them beforehand. Thus, companies that intend to
introduce agility should be aware of the knowledge that is available and this is not always the case in Chile. However, there are new situations that are not so widely reported in scientific literature, such as those related to the particular characteristics of the Chilean software industry. We hope that these new findings can be useful for other companies intending to introduce agile practices into their software development process when their context is similar. Also, for those companies where some practices did not work well, this research can provide them with a means for adapting them so that they could still be applied. We also foresee the adaptability of the metric defined and its thresholds as a means for measuring process improvement.

As a consequence of the results obtained in TSol we are currently analyzing the relationship between context factors and agile practices that are applicable in order to build a framework that may guide in implementing hybrid software processes.

ACKNOWLEDGEMENTS

The work of Jacqueline Marín was partly supported by CONICYT-PCHA/Doctorado Nacional/2016-21160112.

REFERENCES