American Journal of Software Engineering and Applications
2014; 3(1): 1-11
Published online April 20, 2014 (http://www.sciencepublishinggroup.com/j/ajsea)
doi: 10.11648/j.ajsea.20140301.11

Do agile methods increase productivity and quality?

Gabriela Robiolo, Daniel Grane

Informatica, Universidad Austral, Buenos Aires, Argentina

Email address:
grobiolo@austral.edu.ar (G. Robiolo), daniel.grane@gmail.com (D. Grane)

To cite this article:
Gabriela Robiolo, Daniel Grane. Do Agile Methods Increase Productivity and Quality? *American Journal of Software Engineering and Applications*. Vol. 3, No. 1, 2014, pp. 1-11. doi: 10.11648/j.ajsea.20140301.11

Abstract: The Agile methods popped up in the history of software development methods as a solution to several frequent problems, but what is still not clear is whether they produce a significant improvement in productivity and quality or not, if they are compared to the traditional software development methods. In order to clarify this issue and contribute to a better understanding of these methods, we designed an empirical study in which Agile and traditional methods were compared in an academic context. By applying a traditional method to the development of software products, we managed to obtain a more reproducible result, though we could not obtain evidence of an improvement in quality. On the contrary, by applying an Agile method, we obtained evidence of higher productivity, but with a significant dispersion, an aspect that would be interesting to analyze in future studies.

Keywords: Agile, Rup, Scrum, Productivity, Quality, Extreme Programming

1. Introduction

In the 1990s, new processes and methodologies that deal with software development projects appeared. As they evolved, and because of their particular characteristics, these development methodologies fell into two broad categories: traditional and Agile. On the one hand, traditional methods involve those in which the systems are fully specified, they are predictable and they are built according to meticulous and extensive planning. Besides, these projects are run by a clearly defined head that controls the activities, based on explicit knowledge. The organization where this happens is usually large, bureaucratic, with a high degree of formalization, which makes communication formal too. In addition, the software life cycle of their products may be described as waterfall or spiral, and the testing is most surely performed at the end of the cycle [1].

On the contrary, Agile methods are based on the premise that high quality software -adaptable to different conditions- is developed by small groups, using a design which gets continuous improvement. Besides, constant testing provides a rapid feedback, thus making the early introduction of improvements possible. The management structure of the organization in which Agile methods are used tends to be informally defined; it is based on natural leadership and collaboration. Regardless of the size of the organization, work is divided into small groups, where communication within the teams is informal, the internal organization is flexible and their members are participatory. The life cycle of their products is often evolutionary, which includes requirements management and continuous testing [2].

In fact, the Agile methods popped up in the history of software development methods as a solution to several frequent problems. The principal ideas of this solution are summarized in the Agile manifesto[1], which states: "We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value: individuals and interactions over processes and tools, working software over comprehensive documentation, customer collaboration over contract negotiation and responding to change over following a plan". Their creators consider that the items in bold letter have more value than the others.

The Agile methods [3] have greatly impacted on the manner in which software is developed worldwide, so it is convenient to learn if the Agile methods have actually improved the software development life cycle, principally in aspects such as productivity and quality. Consequently, we would like to answer the following research question: *do the use of Agile methods necessarily lead to improved productivity and quality, if compared to those obtained by traditional methods, or is it that the new methods are just an*

[1] http://agilemanifesto.org/

evolution of the traditional ones, without a significant impact on productivity and quality, as observed by Hirsch [4]?

In order to clarify this issue and contribute to a better understanding of these methods by providing more empirical evidence, we designed an empirical experience. This empirical study compared similar applications which were developed by applying either the Rational Unified Process (RUP) , SCRUM or Extreme Programming (XP) in an academic environment. RUP was selected because it is a framework that merges different development methodologies and because due to its characteristics it may be considered a traditional method. SCRUM was chosen because it is popular in the industry [3] and XP because it is the Agile method for which more empirical results have been reported [3].

In the coming sections of this paper a collection of related articles will be commented; the planning, the execution and the results of our empirical experience will be described; such results will be discussed and finally, conclusions will be drawn.

2. Related Work

There has been increasing interest in empirical studies concerning Agile methods for some years [5]. In fact, several empirical studies were conducted in either industrial or academic environments, for which different techniques were employed: formal experiment, survey, case study or post mortem analysis. Lately, the interest to perform objective comparisons has especially increased, principally in aspects such as productivity and quality.

For example, Dyba and Dingsoyr [3] conducted a systematic review of the Agile development methods. Their review points out that three out of the four studies that addressed the comparison of the productivity obtained by Agile and traditional teams found out that using eXtreme Programming (XP) resulted in increased productivity in terms of LOC/h. Also, another study that focused on the use of SCRUM in a very small company [6], which was not included in the previous systematic review, reached, using the same unit of measurement, the same conclusion as regards productivity. However, we may argue that LOC is not an appropriate measure to ensure an unbiased comparison when comparing productivity because experienced programmers have the capacity to summarize in a short statement what novice programmers write in several lines, thus the latter seem to yield a higher production. Nevertheless, we also found out that a later systematic review based on twenty eight very good papers [7] found evidence of the increase in productivity when using the SCRUM method.

Regarding product quality, most studies in Dyba and Dingsoyr's survey [3] reported increased code quality when Agile methods were used. However, none of these studies had an appropriate recruitment strategy to ensure an unbiased comparison, and few quantitative measurements were made, so there seemed to be little scientific support at that moment to claim such improvement in code quality. In addition, Sfetsos and Stamelos [8] conducted a survey on Agile projects in which internal and external quality -based on the ISO/IEC 9126 standard, which is the same standard used in this article- were evaluated. The survey presented forty six high quality empirical articles, twenty seven of which had been developed in academic contexts. All the articles of the industrial context, but one, described cases in which Test-driven Development was applied, while Test-first Development was used in the academic context. The authors reported an improvement in external quality, measured in terms of the number of defects and successful external testing, but there was not the same evidence for the internal quality of either Test-driven Development or Test-first Development. Regrettably, due to the characteristics of our study, it was not possible for us to show evidence of improvement in external quality and our conclusion on internal quality is similar to theirs.

In addition, some studies have been made on the productivity of geographically distributed development. For example, Sutherland et al. [9] reported that Xebia -a Dutch company- started local projects with teams composed of Dutch and Indian members. After obtaining local hyper-productivity in the performance of a team working with SCRUM in the Netherlands, they moved the Indian members of that team into India. Their work in India, also with SCRUM, was as productive as that in the Netherlands. Based on this experience, Xebia has set a model for high performance, distributed, offshore teams, which have obtained one of the lowest defect rates in the industry. Although their hyper-productive performance was defined in terms of the comparison with only one external project, it was an interesting example of geographically distributed development. Another example of geographically distributed development using SCRUM was reported, but this time, a team of 4 persons did not get a significant productivity improvement, if compared to the previous phase of their project, in which a waterfall process had been applied [10].

Besides, there was a longitudinal industrial study which investigated the effects of SCRUM on software quality -in terms of defects and defect density-, and studied the quality assurance process [11]. The authors reported that they followed a project over a three-year period; they used a plan-driven process to compare the software quality assurance processes and software defects of such project during a 17-month phase, which was then followed by a 20-month phase, during which they used SCRUM to make such comparisons. The results of the study did not show a significant reduction of defect densities or changes of defect profiles after SCRUM was used. Likewise, the same conclusion had already been reached [6] in the context of a very small company. Also, Hashmi and Baik [13] had compared XP to a traditional method which was based on the Spiral model. They did not find a significant difference in the quality measured in Fault Rate (Faults/KLOC).

Moreover, Mirakhorli et al. [14] applied a RDP technique - an XP customization method – to the second version of the Union Catalogue System – a virtual catalogue coordinated by the National Library of Iran-. They reported that higher quality and productivity results may be obtained if XP practices are tailored, considering their project sizes, contexts and capabilities.

Finally, it is interesting to mention that Goldin and Rudahl [15] performed a comparative study (RUP versus XP) in circumstances similar to those in which our empirical experience was conducted. They found out that all the teams understood their assigned processes, but the RUP teams were more successful in applying the method. However, the RUP teams were significantly more likely to say that they would have preferred to use XP rather than their assigned process, which is exactly what was said by the students that participated in our empirical study. Nevertheless, the quantitative measurements focused on software processes did not show clear results in terms of productivity and quality. Another interesting article we should highlight is one which deals with a different approach: the use of a hybrid process which took some characteristics from RUP and others from SCRUM. In this case, the requirements and architectural specifications were written following the RUP method, and the Programming, Testing and Deployment were performed according to the SCRUM method. The authors reported an improvement in productivity when this hybrid method was applied [16].

To conclude, the related work described above shows an increase in productivity and external quality when using Agile methods but, as regards internal quality, there is no evidence of such an improvement, which is similar to what is being reported by our empirical study.

3. Empirical Study

Our empirical study was developed in the context of a design workshop that is part of the curriculum of the Software Engineering degree offered by the School of Engineering of Universidad Austral (Argentina). We followed the recommendations of [17, 18] to develop this empirical experience. Besides, to present this study and its replications, the guidelines for reporting empirical research in software engineering in [19] were followed as closely as possible.

We will now present the planning of our study, its execution, and the results we obtained, with an explanation of the threats to validity. Finally, there will be a discussion of the results obtained.

3.1. Planning

The planning stage of our empirical study will be presented by defining our goal, explaining how the study was designed and describing the characteristics of its execution.

3.1.1. Goal

The goal of our empirical study was to make an empirical evaluation of how significant the improvement in productivity and quality may be when applying Agile software development methods, if such results are compared to those obtained when using traditional methods.

To clarify this goal, the Goal-Question-metric paradigm was applied [20]. Table 1 shows the results of its application, where the measures used to compare the methods are listed.

It is important to highlight that the ISO/IEC 9126 standard [21] defines Usability in terms of five sub-characteristics: Understandability, Learnability, Operability, Attractiveness, and Usability Compliance. Usability Compliance refers to the capability of the software component to adhere to standards, conventions, style guides and regulations relating to Usability. In our study, these characteristics were evaluated by asking a set of questions of the users.

In order to measure the Maintainability of an application, Parnas [22] introduced the idea of considering the number of affected modules when a change is proposed. Chaumun et al. [1] assessed the changeability of an object-oriented system by computing the impact of the changes made to the classes of the system. We applied the same concept, but in a simpler manner: every class that was modified by a change was counted. The only exception was the addition of a sub-class, as this was considered an extension of the functionality of the class and, due to the advantages of polymorphism, the pre-existing code was not modified.

Table 1. *Goal-Question-metric paradigm application.*

Questions	Answer	Metric
Which are the most representative Agile methods?	SCRUM and XP are the Agile methods that yield more empirical evidence, which facilitates the comparison of the results [9].	--
Which are the traditional methods?	RUP is a unified method that has all the characteristics of a traditional method.	--
Which are the characteristics and sub-characteristics of quality defined in the ISO/IEC 9126-1 standard that are relevant for the comparison of the selected methods?	Functionality: Accuracy Usability: Understandability, Learnability, Operability and Attractiveness Maintainability: Changeability	Number of Failures reported Degree of usability Number of classes modified when a requirement change is made Degree of understanding of the design
What is productivity in a software development project?	Productivity = Size/Effort	Number of Transactions

Another aspect that may affect the maintenance task is the Degree of understanding of the design (DUD). In our study, this was measured by a set of questions which the developers were asked to answer. In a similar manner, Deligiannis et al [23] included a set of questions in an empirical investigation, as a complement of the maintainability test, which captures the participant's personal opinions regarding the architectural aspects of a system, as well as about the modification tasks.

The main difference between this test and ours is that the participants of their empirical investigation did not develop the product, they only evaluated it.

Productivity was also an important aspect to be considered. As all the projects in our study took the same time to be developed, to compare the productivity of the projects, we only had to consider size. To measure size, the concept Transaction (T) [24] was used, but in this case, it was applied to the final implemented product. Each transaction was identified from the stimulus triggered by the actor into the system, so the functional size was calculated as the number of stimuli in the final product. It is important to note that a T "transaction" has a finer granularity, if it is compared to that of a Function Point (FP) transaction; a FP transaction may be equal to one or more T "transactions". Besides, T has the advantage that it may be applied to game applications.

3.1.2. Design

To measure productivity and quality improvement in software development, we decided to divide our advanced students into seven teams which had to develop a product, using a given developing method in a limited time.

To form the groups, there were two alternatives: either to randomly select members or to form groups of members who had similar capabilities. Although all the students were advanced, their levels of performance were different, so the second strategy was adopted to form balanced groups. To evenly distribute the people into the groups, the following parameters were considered:

(1) Academic performance: the final mark the students had obtained in the prerequisite course was considered.

(2) Experience: number of months they had worked in the industry or in software development labs.

(3) Academic workload: number of courses being attended at that moment and the number of final exams each student still had to sit for.

The developing method - RUP, SCRUM or XP- was randomly assigned to each team.

RUP is the framework which resulted from the unification of different approaches to software development, including the use of UML. It is iterative, incremental, architecture-focused and based on use cases. The process is organized into four phases: initiation, development, construction, transition, and into five processes: requirements capture, analysis, design, implementation and testing. It provides a disciplined approach to defining roles, activities and deliverables. It can be used in large or small organizations, and in formal or informal ones. Besides, it can be used with different management styles because this approach is flexible [25].

On the other hand, SCRUM is an Agile method which is focused on project management. When using this method, software development is performed by a group, during time intervals called "sprints". Each sprint will produce a product increment, which will start with the sprint planning and end with the product increment review. The main roles in SCRUM are: the "SCRUM master", who is the supplier of the process, the "Product Owner", who represents the stakeholders and the business, and the "Team", which is a dynamic and self organized group of about seven people who do the developing task. The Product owner defines the set of requirements that should be implemented, which defines the product "backlog". The Product Owner gives priority to the different requirements and the team determines which of such requirements may be completed during the next sprint, and records this in the sprint backlog. Group members coordinate their work during a daily stand up meeting [26].

Likewise, XP [14] is an Agile method that is defined by a set of rules which characterize it. These rules may be summarized as follows: continuous testing, clearness and quality of codes, common vocabulary, authority to be shared by everybody and at least two people have the understanding necessary to do any task, Test-First Programming is done in pairs.

Table 2 summarizes the main differences between RUP and the Agile methods.

The professor who ran the workshop designed the tasks to be performed by the students. Different tasks were planned for each type of method, RUP or Agile, as presented in Table 3. It is important to note that all the students had to use a Vision Report [12] and a requirements definition, no matter which method they used.

In order to measure the differences between products, we selected the following variables: accuracy, usability, changeability and functionality. Table 4 shows the *variables* involved in the empirical experience and the measure used to measure them. We considered it was necessary to control the following co-factors:

(1) Development environment: all groups worked in similar development environments and used computers with similar specifications.

(2) Time: all the groups worked during the set time.

(3) Level of training: all the students had received similar initial training in each specific topic, and those students who had previous experience in each specific topic were distributed in a balanced way.

(4) Product complexity: the professors controlled the complexity of each product in order to prevent distortions in the developed products. For example, for the game product, a set of complexity rules was defined.

Once the study had been designed, we wrote the following research questions, whose answers would tell us if

there are differences in productivity and quality when using RUP and Agile methods:

(1) Is the number of F reported from the application of an Agile method bigger than that obtained when using the RUP method?

(2) Is the DU analyzed for RUP greater than that in Agile?

(3) Is the number of MC resulting from the application of an Agile method bigger than that obtained when using the RUP method?

(4) Is the DUD in RUP team members greater than that in Agile team members?

(5) Is the size, measured in number of T, of a final product developed with an Agile method bigger than that obtained with a RUP method?

***Table 2**. Comparison between RUP and the Agile methods.*

Aspect	RUP	Agile Methods
Client	Defines and approves the requirements. Validates the system.	Integrates the development team. Defines priorities.
Strength of the work group	Lies on the process	Lies on the people
Architecture	Architecture centered	Gives importance to code
Requeriments	Use cases	The method to be applied is not explicitly stated, but the method most widely used is User Stories
Documentation	Consists of an adequate selection of artifacts	Included in the code
Testing	Is a discipline	The automated test is essential, it completes the requirements definition
Project management	Is a discipline	The importance of this aspect is not explicitly defined, but results show that when applying the Agile methods, the management control improves
Project size	Small, medium and large	Software release
Team size	Not defined	About 10 people

***Table 3.** Tasks to be performed/ artifacts to be developed.*

Method	Task
RUP	Artifacts developed (Vision Report, Use cases, Class Diagram, Sequential Diagrams, External design, Design of test cases) Programming and testing
SCRUM and XP	Artifacts developed (Vision Report, Users Stories) Programming and testing

***Table 4.** Variables measured.*

Variables	Measure	Comment
Accuracy	Number of Failures reported (F)	
Usability	Degree of usability (DU)	Number of very low, low, acceptable, high, and very high answers
Changeability	Number of classes modified when a requirement change was made (MC)	The anonymously nested classes implemented in Java were not taken into account
	Degree of understanding of the design (DUD)	Number of clear, confusing and misleading answers
Functional size	Number of Transactions (T)	Number of stimuli dispatched from the actor to the system, measured in the final product

3.2. Execution

Advanced students, who were the *experimental subjects*, were divided into seven groups: 3 used RUP, 1 SCRUM, and 3 XP. These projects were developed in a four-year period; not all of them were done at the same time. Table 5 shows the capabilities of the *experimental subjects*.

The descriptions of the developed products, i.e. the *experimental objects*, are shown in Table 6.

The professors played the role of leaders, owners and clients. The only exception was P3, in which the client role was played by the students. In every product the students played the role of developers. Table 7 shows the roles played.

The projects were developed by the students during an academic year, at the end of which, the students and professors measured the following measures in the context of a final assessment:

(1) CMC: a set of changes to be made to their final product was defined by the professors. Students examined the changes and identified the class that would be affected by these changes.

(2) DUD: the students involved in the empirical experience answered a set of questions. The product characteristics were considered to design such questions.

(3) T: the professors measured the final products.

The measures F and DU were discarded. As regards the first variable, it was found out that the products had been developed up to a level in which no failures had been reported since, prior to delivery, the products had been tested and the errors corrected. The second variable was ruled out because of the limitations imposed to keep the complexity of the products at a comparable level, which obliged the participants to develop products of similar external designs.

Table 8 shows the changes proposed to measure MC and Table 9 shows the questions made to evaluate the DUD of the product.

Table 10 shows the number of weeks set per task. All the products used a Vision Report [12] and a use case description for requirements definition or user stories. P1

and P2 used the same Vision Report and Use Case textual description.

As an example, and in order to highlight the differences between Agile and traditional methods, Fig. 1 shows a comparison of the use of time made by $P2_{SCRUM}$ and $P1_{RUP}$.

3.3. Results

The values obtained when the above mentioned variables were measured are shown in the following sub-sections.

Table 5. *Experimental subjects' capability.*

Member of group	Final mark obtained in prerequisite course	Work experience (in months)	Number of pending final exams	Courses being attended
$P1_1$	9.5	3	1	7
$P1_2$	4	12	5	6
$P1_3$	9	4	1	7
$P2_1$	7	5	6	8
$P2_2$	6.5	0	5	7
$P2_3$	7.5	5	7	5
$P2_4$	8.5	12	4	6
$P3_1$	6.5	0	0	7
$P3_2$	6	0	1	8
$P3_3$	4	0	3	7
$P3_4$	8	0	0	7
$P4_1$	6	0	5	6
$P4_2$	7	0	5	7
$P4_3$	6.5	3	2	7
$P5_1$	8	12	0	7
$P5_2$	7	3	1	7
$P5_3$	7	6	1	9
$P5_4$	7	0	0	7
$P6_1$	5	8	9	4
$P6_2$	4	7	4	6
$P6_3$	8	7	1	7
$P6_4$	5	6	1	9
$P7_1$	7	10	0	8
$P7_2$	6	18	8	5
$P7_3$	6	0	4	7
$P7_4$	8	0	0	7

Table 6. *Description of the experimental objects.*

Product	Method	Year	Description
P1	RUP	2008	Turn-based strategy game
P2	SCRUM	2008	Turn-based strategy game
P3	XP	2010	Social network
P4	RUP	2010	Social network
P5	XP	2011	3D Social network
P6	RUP	2011	Social network
P7	XP	2011	Social network

Table 7*. Roles played*

Method	Role	Performer
P1, P4 and P6	Team leader	Professor
P1, P4 and P6	Developers	Students
P1	Client	Professor
P4 and P6	Client	Students
P2	SCRUM master	Professor
P2	Developers	Students
P2	Owner-Client	Professor
P3, P5, P7	Clients	Students
P3, P5, P7	Owner	Professor
P3, P5, P7	Developers	Students

Table 8*. Proposed changes.*

Project	Proposed Change
P1 and P2	Place more than one troop in a square. Assign multiple improvements to the troops. Create a new special unit that has the ability to build settlements. Incorporate multiple end-game conditions. Insert a stage of buying and selling resources between shifts.
P3	Make the system limit the number of users that can answer a survey, and do not allow the user to modify such limit. Make the system limit the number of users that can answer a survey, but allow the user to modify such limit. Reject the answer given to a survey. Verify that no poll with the same name has been created before. Unsubscribe fake users (SPAM). Vary the condition to accept a survey.
P4	Enable more than one person to own a site. Unsubscribe fake users (SPAM) and cleanse the system of their activities Add a new strategy to recommend sites. Divide the system into regions. Incorporate a contact address book and invite your contacts to contact you.
P5	Add a new site from a cell phone, according to current geographical location. Add an avatar/person to the world (3 avatars/persons). Visit your friend's home when she/he is present. Add a characteristic to an avatar/ person (run). Add a new scenario.
P6	Add a new type of message for a sub-set of your friends. Add a new filter. Show the historical list of messages from your friends. Query messages using several categories. Chat (direct messages). Query tweets sent exactly a year ago.
P7	Add the right of admission. Notify an interest group of the news about a certain type of event. Create an event for a specific family. Add a chat facility. Use the same system to organize a football tournament.

3.3.1. Changeability

Table 11 shows the number of classes affected by the changes outlined in the previous section, in Table 8. The mean was 9 MC for the RUP-developed projects and the

mean standard deviation was 1 MC. In the case of Agile methods, the mean was 11.25 MC and the standard deviation was 3.86. The differences between the values obtained for the RUP and Agile methods were not significant, with the exception of P2 -which had been made with SCRUM-, which proved to be the weakest design.

Table 12 shows the responses to the questions listed in Table 6. Responses were classified into clear, confusing and misleading. The P4 group did not respond all the questions because there was not enough time to do so, so the professor selected only some questions to be asked of each of those students.

Table 13 shows the statistical analysis of the responses. The mean of the Agile clear responses was bigger than that of the RUP responses. The standard deviation of the Agile responses was similar to that of the RUP responses in the case of the clear responses, while they were bigger than those of RUP's in the case of the confusing and misleading responses. In any case, these differences did not show a significant difference in the degree of understanding of the designs.

Table 9. *Questions to evaluate the degree of understanding of the product design.*

Project	Questions
P1 and P2	How does the system implement the take over of resources? How does the system implement the take over of a source of resources? How does the system implement the movement of troops? How does the system implement the game over? How does the system implement a combination of troops? How does the system implement an attack? How does the system implement the construction of a troop? How does the system implement improvements in a troop? How does the system implement the exploration of mysterious places? How does the system implement improvements in a settlement? How does the system implement the creation of a new game? How does the system implement the completion of a game? How does the system implement the online upgrade of the game? How does the system implement the visualization and rendering of the map? How does the system implement the overall control of the game?
P3	How does the user accept a notification? How does the user add a survey? How does the system add a user? How is a user deleted? How is a survey answered? How is the profile of an owner defined, as opposed to that of a visitor? What design pattern was used in the Qnet implementation? What is the criterion for listing surveys? How does the connection to the database work? What are the layers of the system? How did I save the users' tags? How is the recommendation of the places made?
P4	How is a discount added? Where are the contents uploaded by users stored? What would happen to the system if there were thousands of comments about only one place? How is a complaint from a place implemented?
P5	How does the system validate a user input? What URL encoding strategies are used? How does the system filter search results? How does the system manage authentication and authorization? How does the system implement the synchronization of the avatar/persons? How does the system implement the connection to the database? What is the avatar world? How does the system implement "the avatars walking in the world"? How does the system implement the "bulletin board"?
P6	How does the system implement the connection to the database? How does the system implement the synchronization with the cellular phone? Describe the API structure. How does the system implement the messages update? How do the clients migrate to other devices?
P7	How does the system implement the connection to the database? How does the system notify of a change in an event? How does the system implement the interest groups? How does the system implement the algorithm that suggests friends? How does the system processes the answers from other users?

Table 10. *Weeks allotted per task.*

Project	Task	Weeks
$P1_{RUP}$	Artifacts development (Vision Report, Use cases, Class Diagram, Sequential Diagrams, External design, Design Test)	13
	Programming	19
$P2_{SCRUM}$	Artifacts development (Vision Report)	5
	Programming	27
$P3_{XP}$	Artifacts development (Vision Report and User stories)	4
	Programming	28
$P4_{RUP}$	Artifacts development (Vision Report, Use cases, Class Diagram, Sequential Diagrams, External design, Design Test)	17
	Programming	15
$P5_{XP}$	Artifacts development (Vision Report and User stories)	5
	Programming	28
$P6_{RUP}$	Artifacts development (Vision Report, Use cases, Class Diagram, Sequential Diagrams, External design, Design Test)	16
	Programming	16
$P7_{XP}$	Artifacts development (Vision Report and User stories)	5
	Programming	28

Table 11. *Number of classes affected by the changes.*

Project	Product	MC
$P1_{RUP}$	P1	8
$P2_{SCRUM}$	P2	17
$P3_{XP}$	P3	9
$P4_{RUP}$	P4	9
$P5_{XP}$	P5	9
$P6_{RUP}$	P6	10
$P7_{XP}$	P7	10

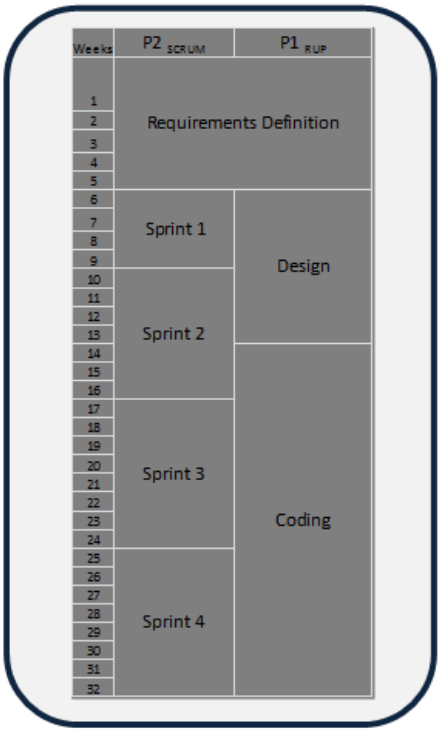

Figure 1. *Comparison of the use of time made by $P2_{SCRUM}$ and $P1_{RUP}$.*

Table 12. *Responses used to determine the degree of understanding of the designs.*

Group	Response		
	clear	confusing	misleading
$P1_{RUP}$	35	4	6
$P2_{SCRUM}$	28	19	13
$P3_{XP}$	39	1	0
$P4_{RUP}$	12	5	0
$P5_{XP}$	14	5	0
$P6_{RUP}$	18	2	0
$P7_{XP}$	18	2	0

Table 13. *Statistical analysis of the responses used to determine the degree of understanding of the designs.*

Statistical Analysis	Response		
	clear	confusing	misleading
RUP mean	21.67	3.67	2.00
RUP standard deviation	11.93	1.53	3.46
Agile mean	24.75	6.75	3.25
Agile standard deviation	11.18	8.34	6.50

3.3.2. Functional Size

Table 14 shows the functional size of each product measured in T. The mean functional size was 7 T for the RUP-developed projects, and the standard deviation was 1 T. For the Agile methods, the mean functional size was 30.5 T, and the standard deviation was 17.82 T. These values are significantly bigger than the RUP values.

Tabla 14*. Functional size.*

Project	Product	Measure [T]
$P1_{RUP}$	P1	8
$P2_{SCRUM}$	P2	10
$P3_{XP}$	P3	23
$P4_{RUP}$	P4	6
$P5_{XP}$	P5	51
$P6_{RUP}$	P6	7
$P7_{XP}$	P7	38

3.4. Answer to Research Questions

After having analyzed our results, we may answer our research questions:

(1) It was not possible to verify if there was a difference in the F reported by the users of Agile and RUP methods, as none reported any failure.

(2) It was not possible to verify if there was a significant difference in the DU analyzed for RUP when compared to that obtained with Agile methods, as the external designs were similar.

(3) There was not a significant difference in the number of MC obtained when applying an Agile method when compared to that obtained when using the RUP method.

(4) There was not a significant difference between the DUD obtained for RUP and Agile, a result that surprised us because RUP is an architecture-centered method.

(5) The size, measured in number of T, of a final product developed with an Agile method was similar to, or bigger than, that resulting of a RUP product.

3.5. Threats to Validity

Four different types of validity will be discussed: internal, external, construct, and conclusion [18].

(1) Internal. Internal validity concerns the cause-effect relationship, that is, if the measured effect is due to changes caused by the researcher or due to some other unknown cause. In this case, it would mean that any measured difference between the applied methods would not be due to the method.

One of the biggest concerns when designing this study was for the products to be obtained to be comparable, that is, that they should have a similar level of complexity. For the game products (P1 and P2), it was necessary to write a set of specific rules in order to avoid non-comparable developments. For example, the time spent on the graphical interface was limited to that required to achieve the minimum necessary level to understand the product, so the products showed similar graphical interfaces. In the case of P3-P7, for which social networks were developed, the biggest difference was the developing environment, which was controlled by the professor, who led the students to the same level of training in every environment. Besides, the applications may be considered comparable, because both methods were used to develop applications of either one of these two types: game product or social network.

Although the groups were formed in a manner as balanced as possible, it is clear that there are personal factors that are

difficult to control. For example, the personal attitude of a person in a certain situation may lead the other members to enhance some of their own personal characteristics, thus resulting in the whole group improving its behavior. Despite this limitation, which is inherent to working with people, the groups had different but comparable behaviors.

(2) External. The external validity of a study describes the possibility to generalize its results. The limited number of projects and measurements does not allow us to generalize our results. However, we think that it was a good experience, which could be replied in academic and industrial software development environments in order to obtain generalized conclusions.

(3) Construct. The construct validity reflects the ability to measure what the researchers are interested in measuring. In this case, the objective was to measure the difference, in terms of productivity and quality, between traditional and Agile methods. It is possible to wonder if the selected variables were suitable to satisfy the purpose of the empirical study. To deal with this limitation, the QGM approach was applied. Also, the measures were selected by giving priority to objectivity and feasibility of measurement, focusing on the internal and external features of the obtained products.

(4) Conclusion. The conclusion validity describes the ability to draw statistically correct conclusions based on measurements. In this experience, the limited available data did not allow us to reach statistically significant conclusions.

4. Discussion

If we want to get an objective idea of the functional size differences between the products developed with the different methods, we have to consider the information we obtained about the weeks spent on programming tasks, which is presented in Table 10, and the characteristics of the persons included in each team, which are described in Table 5. The groups were made up by three or four persons and the best possible combination was sought for, within the restrictions that we had. Also, in these beginner programming teams, we could see that the more people there were on a team, the more coordination problems they had.

The Agile teams began to work in programing tasks no later than in the fifth week, but the RUP teams began to do so between the fifteenth and nineteenth week. So, the Agile groups worked in programming almost twice the time the RUP groups did, but they produced products which had very different functional sizes: from almost similar to that of a RUP product, to three times, or even seven times, bigger than that of a RUP product. We noticed that this increment in productivity was a consequence of the student s' involvement in the planning, estimation and control activities in each sprint. This practice reinforced the commitment and the responsibility of each student and favored the leaders' development, which also contributed to motivate the team.

This shows that Agile groups are usually more productive, but their outcome may have a bigger standard deviation.

One factor that could have produced the dispersion in Agile productivity is pair programming. We believe productivity could be increased by pair programming in XP teams. However, it is important to note that although the premise the students had been given was always to work with pair programming, in fact, they only worked in pairs when the nature of the task justified this type of work. And of course, they did not work in pairs when someone was absent, or delayed, or if for some specific reason, someone worked at home. This type of behavior was also observed by Zazworka et al. [27], so we may conclude that although pair programming is the best option to work, it has to be applied in a flexible manner. Besides, it depends on the persons involved in the task to be done; some people enjoy working in pairs, while others do not.

Moreover, there may be other causes that may explain Agile productivity dispersion. For example, it would be interesting to measure, in order to deeply understand, how motivation may affect the development of a project. In our study, it may have been revealing to learn about the students' and leaders' commitment and motivation, as well as about the leaders' experience in development.

As regards quality, it was not possible to identify significant differences in accuracy, usability and changeability. Actually, the fact that no failures were reported and that the products had similar degrees of usability was a consequence of the conditions we set for the students' products to comply with the academic requisites.

Particularly, it was surprising to see that the Agile methods did not improve changeability, something which is claimed by the developers that use these methods. The reason for this may be that in the context of the ISO/IEC 9126-1 [21] standard, changeability is one of the subcharacteristics of maintainability, not the response given to a client when he/she proposes a requirement change. In fact, we defined the measures MC and DUD to measure the changeability that affects the maintenance phase of a product. So the analysis of the results of such measurement shows there will not be significant differences in the future life of a product, whether a RUP or an Agile method is applied, which is ,in fact, an interesting conclusion.

We may wonder if the selection of the three quality variables was appropriate, as we did not obtain a significant difference when applying the different methods, either RUP or Agile. While planning the empirical study we did not realize that the conditions defined in order to accept the products would not contribute to clarify the differences that the use of these methods would bring about regarding two of the quality characteristics -accuracy and usability-. So, we may suggest not including these two variables in future replications in academic contexts. However, in spite of the fact that the variable changeability was well selected and measured, this sub characteristic is not enough to evaluate the quality an industrial product. Reliability, portability, and efficiency may be key aspects for the marketing of a

software product and it would have been interesting to consider them. To conclude, we have not found evidence in an academic context that Agile methods improve quality characteristics, but this conclusion does not necessarily apply to an industrial context.

Finally, to answer the question: do Agile methods increase productivity and quality?, we may say that in the context of our empirical study, they significantly increased productivity, but not quality. In our case, the circumstances that affected productivity were the time allotted to programing tasks, the application of pair programming practices, the planning, estimation and control practices and the leadership growth. On the other hand, quality was not improved by the Agile methods because they do not introduce practices that differ from those of traditional methods; in fact, the quality results were affected by the people involved in each team, the circumstances defined in order to make the products comparable and the limitations of the academic context in which the study was developed.

5. Final Conclusion

This empirical study, conducted in an academic environment, has helped us understand how the selection of a traditional or an Agile software development method may impact on the productivity and quality of a software project. By applying a traditional method, we managed to obtain a more reproducible result, but we could not obtain proof of an improvement in quality. On the other hand, in our study there was evidence about obtaining higher productivity by using Agile methods. However, it would be recommendable to analyze the circumstances that produced the difference in productivity with the Agile methods, focusing our analysis on the motivation and commitment of the developers and leaders, and on the leaders' experience.

In the future, it would be recommendable to replicate this study in an industrial environment, where junior and senior developers may work together, and to evaluate in a longer period of time if Agile methods lead to increased productivity and quality in software development.

Acknowledgments

Our thanks to the Research Fund of Austral University, which made this study possible.

References

[1] M. A. Chaumun, H. Kabaili, R. K. Keller and F. Lustman, A Change Impact Model for Changeability Assessment in Object-Oriented Software Systems, csmr, *Third European Conference on Software Maintenance and Reengineering*, (1999), pp.130.

[2] S. Nerur, R. Mahapatra and G. Mangalaraj, Challenges of migrating to agile methodologies. *Commun. ACM* 48, 5 (May 2005), 72-78. DOI=10.1145/1060710.1060712 http://doi.acm.org/10.1145/1060710.1060712.

[3] T. Dyba and T. Dingsøyr: Empirical Studies of Agile Software Development: A Systematic Review, *Inform. Softw. Technol* (2008).

[4] M. Hirsch, Moving from a plan driven culture to agile development, in *Proceedings of the 27th international Conference on Software Engineering* (ICSE '05) (St. Louis, MO, USA, May 15 - 21, 2005). ACM Press, NY, 38-38.

[5] T. Dingsøyr, S. Nerur, V. Balijepally, N. Brede Moe, A decade of agile methodologies: Towards explaining agile software development, *Journal of Systems and Software*, Volume 85, Issue 6, (June 2012), pp. 1213-1221, ISSN 0164-1212, http://dx.doi.org/10.1016/j.jss.2012.02.033.

[6] E. Caballero, J.A. Calvo-Manzano and T. San Feliu, Introducing Scrum in a Very Small Enterprise: A Productivity and Quality Analysis, *Systems, Software and Service Process Improvement, Communications in Computer and Information Science* Volume 172, (2011), pp. 215-224.

[7] E. S. F. Cardozo, J. B. F. Araújo Neto, A. Barza, A. C. C. França, and F. Q. B. da Silva, SCRUM and productivity in software projects: a systematic literature review, in *Proceedings of the 14th international conference on Evaluation and Assessment in Software Engineering* (EASE'10), Mark Turner and Mahmood Niazi (Eds.), British Computer Society (Swinton, UK, UK, 2010),131-134.

[8] P. Sfetsos and I.Stamelos, Empirical Studies on Quality in Agile Practices: A Systematic Literature Review, Quality of Information and Communications Technology (QUATIC), 2010 Seventh International Conference, (Porto, Sept. 29 2010-Oct. 2 2010),44 – 53.

[9] J. Sutherland, G. Schoonheim, M. Rijk, Fully Distributed Scrum: Replicating Local Productivity and Quality with Offshore Teams, *Hawaii International Conference on System Sciences* (2009), pp. 1-8, 42.

[10] L. Lavazza, S. Morasca, D. Taibi and D. Tosi, Applying SCRUM in an OSS Development Process: An Empirical Evaluation, *Agile Processes in Software Engineering and Extreme Programming, Lecture Notes in Business Information Processing* Volume 48, 2010, pp 147-159.

[11] J. Li, N. B. Moe, and T. Dyba˚, Transition from a plan-driven process to Scrum: a longitudinal case study on software quality, in *Proceedings of the 2010 ACM-IEEE International Symposium on Empirical Software Engineering and Measurement* (ESEM '10), ACM Article 13, (New York, NY, USA, 2010), 10 pages.

[12] K. Bittner and I. Spence, *Use Case modeling* (Addison-Wesley, 2003).

[13] S.I. Hashmi, J.Baik, Software Quality Assurance in XP and Spiral - A Comparative Study International, *Conference on Computational Science and its Applications*, (ICCSA 2007), (2007), 367 - 374

[14] M. Mirakhorli, A. Khanipour Rad, F. Shams, M. Pazoki, and A. Mirakhorli, RDP technique: a practice to customize xp, In *Proceedings of the 2008 international workshop on Scrutinizing agile practices or shoot-out at the agile corral* (APOS '08), ACM, (New York, NY, USA, 2008), 23-32.

[15] S.E Goldin and K.T Rudahl, Software process in the classroom: A comparative study, *9th International Symposium on Communications and Information Technology* (ISCIT 2009), (28-30 Sept. 2009), 427 – 431.

[16] W.C. de Souza Carvalho, P.F. Rosa, M. dos Santos Soares and M.A. Teixeira da Cunha Junior, A Comparative Analysis of the Agile and Traditional Software Development Processes Productivity, *Computer Science Society (SCCC), 2011 30th International Conference of the Chilean*, (2011), 74 – 82.

[17] N. Juristo, and A.M. Moreno, *Basics of Software Engineering Experimentation*, Kluwer Academic Publishers, 2001.

[18] C.Wohlin, P.Runeson, M.Höst, M.C.Ohlsson, B. Regnell, and A. Wesslen, *Experimentation in Software Engineering: an Introduction*, Kluwer Academic Publisher (2000).

[19] Jedlitschka, M. Ciolkowoski, D. Pfahl, Reporting Experiments in Software Engineering, *In Guide to Advanced Empirical Software Engineering* (2008).

[20] Basili, G. Caldiera and D. Rombach, *The Goal Question Metrics Approach*, Encyclopedia of Software, Engineering, Wiley, 1994.

[21] ISO/IEC, ISO/IEC 9126-1 Software engineering- Product quality- Part 1: Quality model, 2001.

[22] Parnas, D.L., On the criteria to be used in decomposing systems into modules, *Communications of the ACM***.** Volume 15, Issue 12 (December 1972) 1053 – 1058

[23] Deligiannis, M. Shepperd, M. Roumeliotis, and I. Stamelos, An empirical investigation of an object-oriented design heuristic for maintainability. *J. Syst. Softw.* 65, 2 (February 2003), 127-139.

[24] G. Robiolo, C. Badano and R. Orosco, Transactions and Paths: two use case based metrics which improve the early effort estimation, *In Proceedings of 3rd Int. Symp. on Empirical SW Engineering and Measurement* (ESEM 2009) (October S., Lake Buena Vista, Florida, 2009), 15-16.

[25] Jacobson and B. Grady, J. Rumbaugh, *The Unified Software Development Process*, (Addison Wesley, 1999)

[26] K. Schwaber, *Agile Project Management with Scrum*. Microsoft Press (2004).

[27] N. Zazworka, K. Stapel, E. Knauss, F. Shull, V. R. Basili, and K. Schneider. Are developers complying with the process: an XP study, in *Proceedings of the 2010 ACM-IEEE International Symposium on Empirical Software Engineering and Measurement* (ESEM '10). ACM, Article 14 (New York, NY, USA, 2010), 10 pages.