CLEI ELECTRONIC JOURNAL, VOLUME 13, NUMBER 1, PAPER 5, APRIL 2010

Synchronous versus Asynchronous interaction between users of two collaborative tools for the production of Use Cases

Carlos M. Fuenzalida
Universidad de Santiago de Chile, Depto. Ingeniería Informática,
Santiago, Chile,
carlosmfuenzalida@gmail.com

and

Héctor B. Antillanca
Universidad de Santiago de Chile, Depto. Ingeniería Informática,
Santiago, Chile,
hantilla@informatica.usach.cl

Abstract

The performance of requirement engineers in the production of Use Cases is analyzed and compared using a collaborative editor in synchronous mode versus one in asynchronous mode, with the objective to learn which mode allows generating better specification documents for specific software. For this purpose some Use Cases were produced under a quasi-experimental approach. There were proposed eleven metrics, used to evaluate the cases generated in both modes, according to some desirable features. The attained results show that the Use Cases asynchronously produced hold a better quality. On the other hand, the synchronously-generated Use Cases take less time in their production than the asynchronously-generated ones.

Keywords: Use Cases, requirements specification, collaborative system, synchronous work, asynchronous work.

1. INTRODUCTION

The objective is to help in one of the first stages of the Software Engineering, getting and specifying software requirements. This stage is predominantly collaborative, so it obtains benefits from the collaborative software tools. The students from the IT Engineering Department, Departamento de Ingeniería Informática (DIINF), of the Universidad de Santiago de Chile (USACH) have developed several collaborative applications ([1], [2], [3], [4]) that help to create and write Use Cases according the Use Cases theory from the OMT++ methodology [10]. Although these applications are functionally useful for their purpose, we asked ourselves the following question: What kind of collaborative application for the writing of Use Cases improves the performance of its participants? One that supports synchronous work or one that supports asynchronous work? We considered that the answer to this question is useful to guide the development of future collaborative tools, especially in the domain of the software requirement specification, with the purpose of supporting the process in a more efficient way.
For this purpose we selected a pair of the collaborative tools available to create and write Use Cases, one supporting synchronous collaborative work and the other one supporting asynchronous collaborative work. A series of experiments was conducted to evaluate and compare the performance of these tools. Eleven metrics were defined to evaluate the quality of the Use Cases. Through a quasi-experimental design, where 12 groups of three persons each participated, 60 Use Cases were generated, 30 of them in synchronous mode and 30 in asynchronous mode. The experiments showed that asynchronous mode generates better-quality cases, but it takes longer to do so.

1.1 Domain of the problem

In order to proceed with this investigation we used knowledge from three related areas in IT: Computer Supported Cooperative Work (CSCW) [6], Collaborative Software Engineering [7] and Requirements Engineering [8]. From CSCW we take the taxonomy of collaborative systems, which classifies them in four categories [9], according to the time-space distribution of its participants, i.e.: synchronous-located system, synchronous-distributed system, asynchronous-located system and asynchronous-distributed system. In particular, this study uses two scenarios: synchronous-distributed and asynchronous-distributed. Collaborative Software Engineering offers the frame to understand the problems the developers have working in a group. Requirements Engineering is the area of Software Engineering in which the problem this investigation deals with is found: the problem of getting and specifying requirements for a future software system. Indeed, the activities performed in this stage of the software development process are mostly collaborative, involving interaction between software developers, designers and the stakeholders, especially the final users.

2. USE CASES

The Use Cases technique is the most used technique in the process of getting and specifying requirements at the moment, and it is adopted by numerous methods for object-oriented software development, such as RUP, ICONIX, OOSE, OMT, OMT++, Booch, ROOM, etc., because it allows a complete vision of the desired functionality in a system.
In this research we adopted the Use Cases technique that OMT++ methodology [10] uses. OMT++ methodology was developed by Ari Jaaksi for Nokia Telecommunications [11] and at the moment is one of the most used in our community due to its simplicity and completeness being used in software design, development and test. A Use Case describes how a system and the final user "do something ". A Use Case in OMT++ exemplifies a possible use for a future system, so it must be written in a language that can be understood both by the people related to the problem to solve, the stakeholders, and software developers. To document an Use Case in OMT++, a field template is defined to describe different aspects of the case; these are: Title, Actors, Objective, Pre-condition, Description, Post-condition and Exceptions.
In particular, the Description includes the Use Case story, which describes “a typical way to use the system”. The exceptions are added to the Description, like some sort of pointer, and they are showed in the Exceptions section. These describe other uses of the system when the normal flow of the case is interrupted. The Pre-condition and Post-condition sections point out the limit conditions for the Use Case, also connecting them to the other Use Cases. The objective shows the purpose the user acting in the Use Case has in mind for the system. The Actors section contains the names of the different user types that interact with the system in that Use Case.
Sometimes the developers of computer programs don’t write down good cases; to solve this Ari Jaaksi proposed a guide to write Use Cases [10], formed by ten rules called the “ten commandments” to write Use Cases which, if followed, allow generation of good quality Use Cases. The proposed rules are the following:

1. Use Cases must specify the most important functional requirements.
2. A Use Case describes something that the designer can be proud of and the client is willing to pay for.
3. A Use Case describes a typical way to use a system, but nothing more.
4. A Use Case is a play.
5. A Use Case has a beginning, a main body and an ending.
6. A Use Case is like an essay written by an elementary school pupil.
7. A Use Case fits in one page.
8. A Use Case is loud and clear.
9. Customers and software designers can sign the Use Case.
10. A Use Case can be used in system development and system testing.

3. PRELIMINARIES

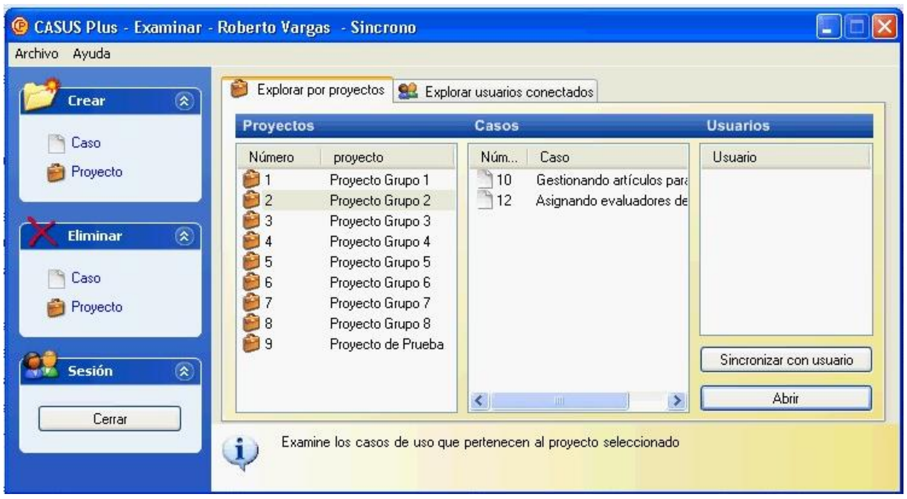
3.1 Preparation of the experiences

In order to perform the experiences two collaborative tools for Use Cases redaction were used, CASUS++ [5] which allows its users to write Use Cases both in synchronous and asynchronous mode; nevertheless, for this study we used only asynchronous mode, and CASUS+ [4] which was specially designed so its users write Use Cases in synchronous mode.
Both tools are functionally similar and their features are showed in Table 1. In Fig. 1 and Fig. 2 we can see the main screens for CASUS+ y CASUS++, respectively; note that both tools have a similar interface.
The template used to write Use Cases in CASUS+ is the one showed in Fig. 3, and the one used in CASUS++ is showed in Fig. 4, both including the same fields to be filled: Title, Actors, Objective, Pre-condition, Description, Post-condition and Exceptions. A substantial difference between these tools is that CASUS+ allows its users to write the contents of that template dynamically, offering for that end a view synchronization mechanism so any user can efficiently access to the view another user is watching and, being that the case, if an user is writing in a shared view, any other user can modify that text online. The implicit supposition for this feature is that the users have the intention to collaborate in the production of the work object, and not to obstruct it. Another difference between the tools is that CASUS++, in asynchronous mode, allows the users to participate in deferred times in Use Cases production through a process structured in 5 stages, where the users must elect a coordinator. The stages are as follows:

1. Stage 1: Users propose Use Cases.
2. Stage 2: Users comment or modify Use Cases proposed by other users.
3. Stage 3: Coordinator summaries Use Cases proposed by the users.
4. Stage 4: Users comment or modify Use Cases summaries proposed by the coordinator.
5. Stage 5: Coordinator writes the final proposal for Use Cases.

Table 1: Features of the collaborative tools

Features	**Collaborative Tools**	
	CASUS+	**CASUS++**
Work mode	Synchronous.	Synchronous and asynchronous.
Use of projects	Yes.	Yes.
Coordinator	No.	Yes in asynchronous mode, not in synchronous mode.
Exporting documents to text files	Yes.	Yes.
Simultaneous text edition	Yes.	Yes.
Collaborative editable field	Only the description field.	Only the description field in synchronous mode.
Communication between participants	Chat and collaborative case description field.	Asynchronous mode: comments and modifications to the cases. Synchronous mode: chat and collaborative case description field.
Programming language	Visual Basic .NET.	Visual Basic .NET.
Platform	Windows.	Windows.
Main contribution of the tool	Improves usability of CASUS (collaborative tool that works in synchronous mode).	Integration of CASUS+ with Asynchronous Collaborative System to get requirements through Use Cases.

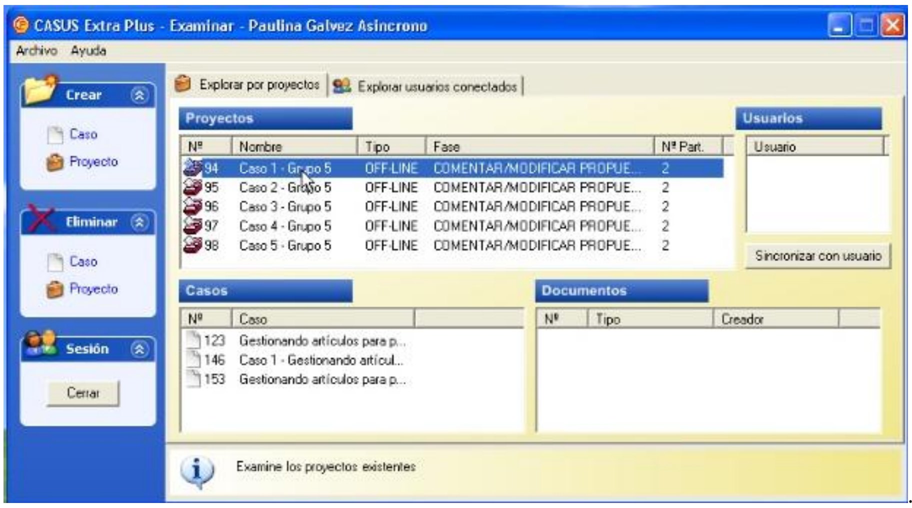


Número	proyecto
1	Proyecto Grupo 1
2	Proyecto Grupo 2
3	Proyecto Grupo 3
4	Proyecto Grupo 4
5	Proyecto Grupo 5
6	Proyecto Grupo 6
7	Proyecto Grupo 7
8	Proyecto Grupo 8
9	Proyecto de Prueba

Núm...	Caso
10	Gestionando artículos para
12	Asignando evaluadores de

Examine los casos de uso que pertenecen al proyecto seleccionado

Figure 1: CASUS+ tool's main screen

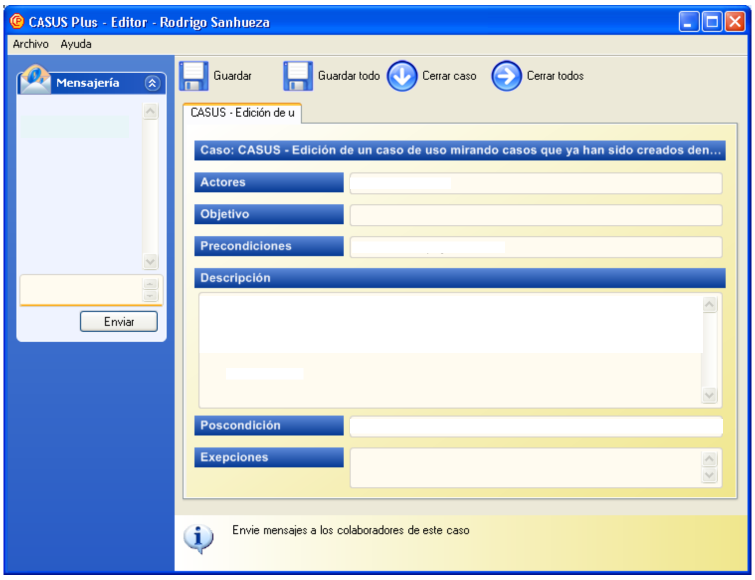


Nº	Nombre	Tipo	Fase	Nº Part.
94	Caso 1 - Grupo 5	OFF-LINE	COMENTAR/MODIFICAR PROPUE...	2
95	Caso 2 - Grupo 5	OFF-LINE	COMENTAR/MODIFICAR PROPUE...	2
96	Caso 3 - Grupo 5	OFF-LINE	COMENTAR/MODIFICAR PROPUE...	2
97	Caso 4 - Grupo 5	OFF-LINE	COMENTAR/MODIFICAR PROPUE...	2
98	Caso 5 - Grupo 5	OFF-LINE	COMENTAR/MODIFICAR PROPUE...	2

Nº	Caso
123	Gestionando artículos para p...
146	Caso 1 - Gestionando artícul...
153	Gestionando artículos para p...

Nº	Tipo	Creador

Figure 2: CASUS++ tool’s main screen



Caso: CASUS - Edición de un caso de uso mirando casos que ya han sido creados den...

Envíe mensajes a los colaboradores de este caso

Figure 3: Use Cases edition screen in CASUS+

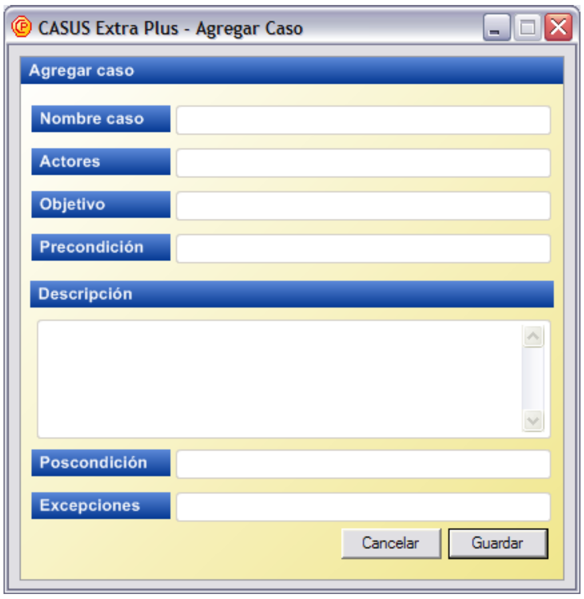


Figure 4: Use Cases edition screen in CASUS++

The experiences of every distinct requirement engineer group were recorded by audiovisual means for their posterior analysis. The Camtasia Studio [12] application was used for this, which allows recording what happens in the computer screen and through external devices, having to use webcams and microphones for every computer.
To know the number of participants necessary for the experiences with the collaborative tools, a sample size was calculated with a reliability of 90% (z = 1.64), a variance (s) of 0.5 (due to the inexistence of previous studies), an error percentage (e) of 15% and a total population size (N) of 48 people, where the only requirement to participate in the experiments was having knowledge and experience in writing Use Cases. We identified this kind of user in senior students of IT engineering; specifically, a requirement was set: that the students had studied and passed the Software Engineering course in the IT Engineering professional career of the USACH according to the equation (1), obtained from [13], 18 people are necessary for every mode, as seen in equation (2). That being the case, for every experience, synchronous and asynchronous, there were a total of 36 students from the DIINF that fulfilled with the requirements for the experiments. Those persons were divided in 12 groups of 3 persons each, 6 for each work mode.

$$n = \left(\frac{z^2 \sigma^2 N}{e^2 N + z^2 \sigma^2} \right) \tag{1}$$

$$n = \frac{1.64^2 \cdot 0.5^2 \cdot 48}{0.15^2 \cdot 48 + 1.64^2 \cdot 0.5^2} \cong 18 \tag{2}$$

3.2 Realization of the experiences

The experiences in synchronous mode were performed with the participants interacting in different classrooms, depending on the available time each group had, using CASUS+. Meanwhile, for the experiences in asynchronous mode CASUS++ was used, and the members of each group used only one classroom in the DIINF, but the participants attended on different hours, so they didn't interact at the same time. In this way each group had the necessary time to redact the Use Cases and concluded the task when they considered the Use Cases were satisfactorily finished.
The groups received a problem formulation for the Use Cases production task, which corresponded with a project for the Software Engineering Projects course given in the DIINF to the fifth year students of the career, who were mostly the ones participating in this experience due to their fulfillment of the established condition, to pass a previous Software Engineering Bases course. The formulation briefly describes the problem, brute functional and non-functional requirements, and proposes 5 Use Cases titles that are taken from the functional requirements. This formulation was delivered to the groups to be written in each one of the experiences.

4. RESULTS

4.1 Metrics to evaluate the Use Cases

Eleven metrics were established to analyze each Use Case generated. Some metrics are based in Ari Jaaski's good writing rules [9]. These metrics are:

1. Client satisfaction grade: it indicates the completeness level of the Use Case description section to describe the necessary activities, according to the client, to achieve the objective of the Use Case. That level is measured in a percentage, where 100% shows a Use Case describing all the necessary activities to achieve its objective. To measure the satisfaction grade, the client makes a list of the activities that he considers necessary to achieve an objective associated to a Use Case, and then that activity list is contrasted with the ones that actually appear in each case and a percentage is obtained.

2. Writing Use Case as a primary school child: it indicates if the Use Case description is written like nine years-old child from elementary school would write a text, as it is said in the sixth good-writing rule for Use Cases. This rule considers that children, not having a well-developed abstraction capacity or a wide vocabulary, express themselves in a simple and straightforward way about facts. The metric is a grade, from 1 to 7, assigned by people with knowledge in Use Cases creation. In this case, tenths are discounted according to the seriousness of the fault the evaluator esteems, for every sentence with explanations, grown-up language or round-about words (which a 9 years-old child wouldn't use), from the highest grade (7.0).

3. Difference between the number of functional requirements described in a Use Case and the functional requirements appearing in a Use Case written by an expert, both written to achieve the same objective

4. Grade assigned by experts in Use Cases writing: indicates the satisfaction degree of the experts in Use Cases redaction regarding the operations it contains. To evaluate a Use Case the number of operations it contains is compared with the number of operations the Use Case written by experts contains. It consists in a grade, from 1 to 7, that indicates the experts' satisfaction level. People with expertise in Use Cases creation, which are the Software Engineering Projects course teacher and his assistant teacher for this case, evaluate each Use Case according to a pattern drawn up to evaluate the cases, assigning them a grade from 1 to 7, where a higher grade means a better case.

5. Word quantity in the Use Case: it indicates if the Use Case adjusts or not to a page size, which is the seventh rule for a good Use Case creation in OMT++. It is measured by the number of words in each case, from the name to the exceptions, and it is compared to the average number of words that fit in an A4 page, in Spanish language. The better the number of words adjusts to this number, the better the case. Use Cases that are too short are also not very useful and they generate too much documentation; Use Cases that are too long tend to be confusing and the documentation becomes unmanageable.

6. Word quantity in the Use Case description section: It indicates the workload dedicated to the Use Case description compared to the one dedicated to the case as a whole. It measures the number of words present in the description section of each case. Knowing this number and comparing it with the word quantity in the Use Case the writing load given to this section of the case, which is more important, can be seen.

7. Text structure of the Use Case: it indicates if the Use Case is well written according to certain structure, i.e., if it has an introduction, main body and conclusion, as the fifth rule of good Use Cases writing dictates. It consists of a grade, assigned by people with knowledge about Use Cases redaction, from 1 to 7. For each fault the case sections have, tenths will be discounted from the grade.

8. Implementation slants quantity: it indicates the number of implementation slants appearing in each Use Case, the more slants in the case, the less quality it has. Implementation slants are features or properties typical of the program implementation, and they shouldn't appear in the Use Cases. For instance, saying in a case that to access to certain place you have to press a button. This metric is expressed as an integer indicating the number of slants.

A series of keywords will be also used as metrics in the Use Cases, which indicate quality for requirement specifications in natural language, as proposed in [14], so the more of these words a case has, the lower its quality. These are:

9. Options words: these words can provoke loss of control over the final product of the application as defined by the Use Case, producing ambiguities and risk of case action tracking. These words are: can may and optionally.

10. Weak phrases: these are phrases that cause uncertainty and could lead to multiple interpretations of the Use Case. The weak phrases reported in [14] are: adequate, as a minimum, as applicable, easy, as appropriate, be able to, be capable, but not limited to, capability of, capability to, effective, if practical, normal, provide for, timely.

A very important metric to considerate is:

11. The time used by the requirement engineers to produce the Use Cases, which reflects the difficulty degree each group had to achieve their objectives. In the case of the synchronous groups the time used by the group to create the Use Case was multiplied by the number of members of each group, while in the case of the asynchronous groups the time each member used to propose and evaluate the Use Cases is added to the total.

4.2 Comparison of the Use Cases

Evaluating and comparing the Use Cases, a higher satisfaction of what was requested by the application client is seen in the Use Cases generated asynchronously, with 12% more satisfaction, as seen in Fig. 5. There are also more functional requirements described in the asynchronous Use Cases, so those cases describe better the functionalities of the future application, requested by the client, must have. To measure the number of functional requirements described in the different cases the rate is calculated dividing the functional requirements described in the Use Case by the number of functional requirements that should appear in the Use Case, so if this rate is 1.0 the case describes all the functional requirements it should, if it is 0.5 only half of them, and so on. In Fig. 6 the rate average for all Use Cases according to the synchronism used is seen.

The writing of the cases written asynchronously is also better than the writing of the ones written synchronously. This is seen in the better writing structure and elementary education child-like writing (easier to understand) in the cases produced asynchronously over the ones produces synchronously, observed both by the software developers and the clients requesting the application. Also, the Use Cases produced asynchronously had a better qualification from experts in Use Cases creation, so their quality is better. The average results of the assigned grades for the different metrics are seen in Fig. 7.

Regarding to ambiguity of the Use Cases produced, the number of options words and weak phrases, which produce uncertainty or variations in the cases interpretation, the asynchronously-written Use Cases show less of these elements than the ones made synchronously, with 15 optional words and 4 weak phrases less, as seen in Fig. 8 and Fig. 9, respectively. It is worth highlighting, though, that in all the generated Use Cases there were found only 10 weak phrases, so the uncertainty number due to those phrases in the cases is low.

The cases generated asynchronously also have less implementation slants than the ones generated synchronously, as seen in Fig. 10, which shows that the requirement engineers could better analyze the cases and not adding details that should be seen in the implementation stage of the future computer program. Thus, these cases are clearer and don't limit the future development of the required tool.

By analyzing the number of words in each case, it shows that the longest ones are the Use Cases written asynchronously, with an average of 13 words more than the produced synchronously. They also lead notoriously in the number of words in each case description; the most important part of the Use Case, because it is where what the case must do is written, with an average of 18 words over the others. Although most of the Use Cases generated in both modes fit in a page, because they have less than 250 words (in Spanish language) in average, the ones generated asynchronously are nearer the limit, without surpassing it, so they adjust better. On the other hand, the difference between the number of written words in the Use Case and the number of words in the average description for the cases written synchronously is larger than the average for cases written asynchronously (see Fig. 11), so the latest are the ones that posses more relevant information in each Use Case.

The greatest advantage the Use Cases generated synchronously had is the production time (hour-man) of the cases. The time it took to generate Use Cases asynchronously was longer than the time used to generate Use Cases synchronously in approximately a 150%, with an average of 2 hours and 42 minutes more, as it is showed in Fig. 12, although if the time a synchronous group was working together and the sum of the times each member of the asynchronous group was working is compared, this difference is the triple.

4.3 Other Results

Through the experiences some behaviors were observed and comments were made by the participants, about the collaborative tool used and the way the members of the group interacted. Many members of the synchronous groups thought that the chat tool CASUS+ has is not good to communicate between them, because some audiovisual mean would be better, using cameras and microphones, for a better understanding, and that it is necessary a way to customize the chat so everyone could know, easily, who is writing something in a given moment. Regarding the collaborative writing screen all the members thought that the customization is also necessary for members, plus a way to control who is writing at a given time.

Regarding the communication between members of each group, in asynchronous mode they could coordinate satisfactorily and they only had some troubles to distinguish who was writing certain text in a certain moment, when doing modifications in the Use Cases. The participants only communicated through comments or modifications made to other's cases, but although they didn't communicate constantly, they got good results nevertheless, thanks to the five stages methodology used, as described in [2].

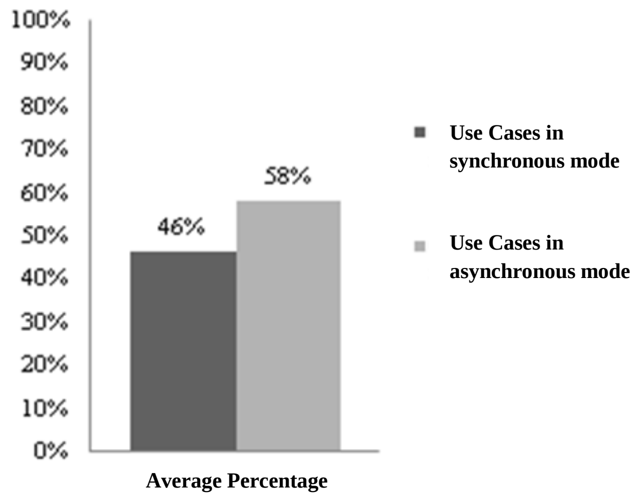


Figure 5: Average percentage of the cases, according to client satisfaction

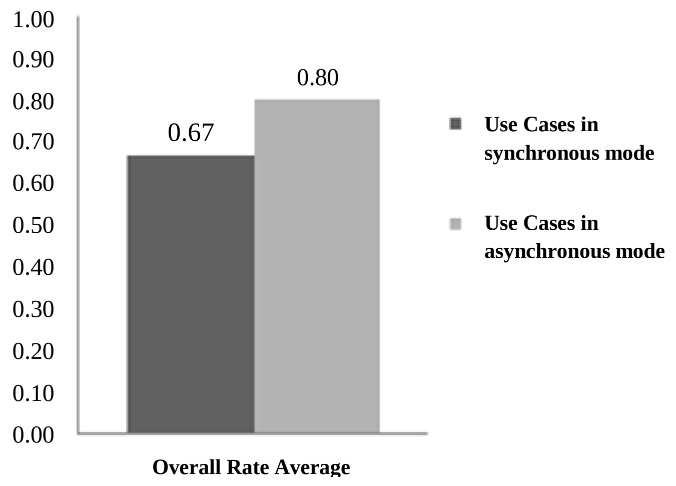


Figure. 6: Rate average of the number of functional requirements described in the Use Cases versus the functional requirement in a Use Case written by an expert

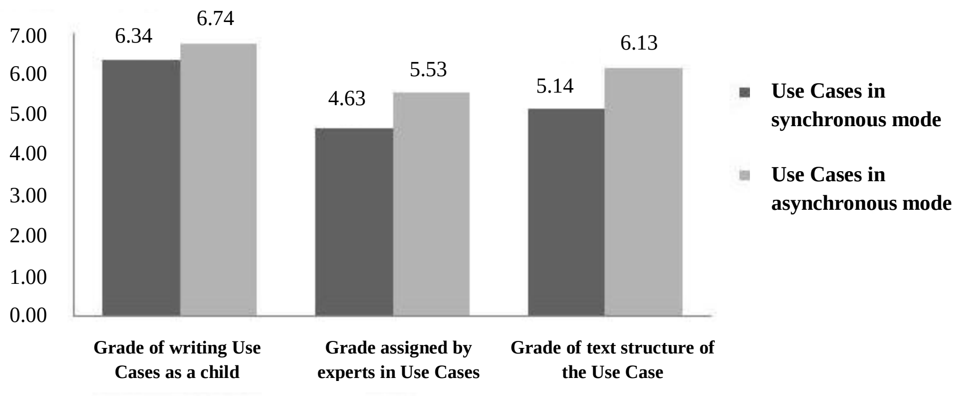


Figure 7: Grade average of the Use Cases, according to three metrics

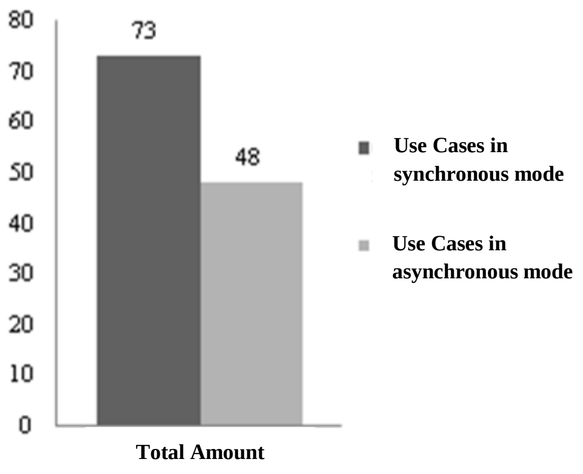


Figure 8: Total optional words quantity in the Use Cases

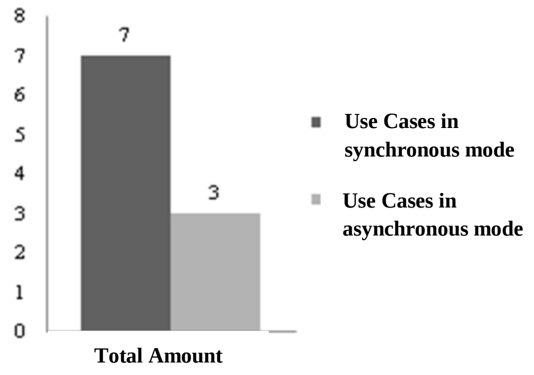


Figure 9: Total weak phrases quantity in the Use Cases

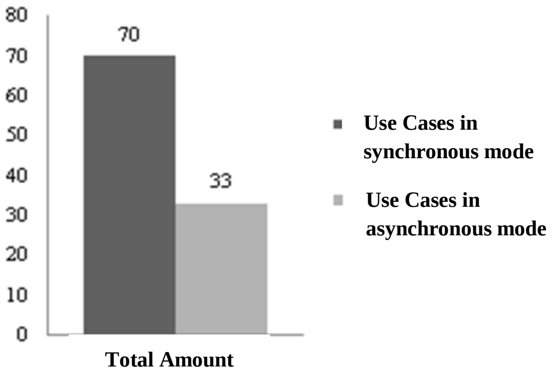


Figure 10: Total implementation slants quantity in the Use Cases

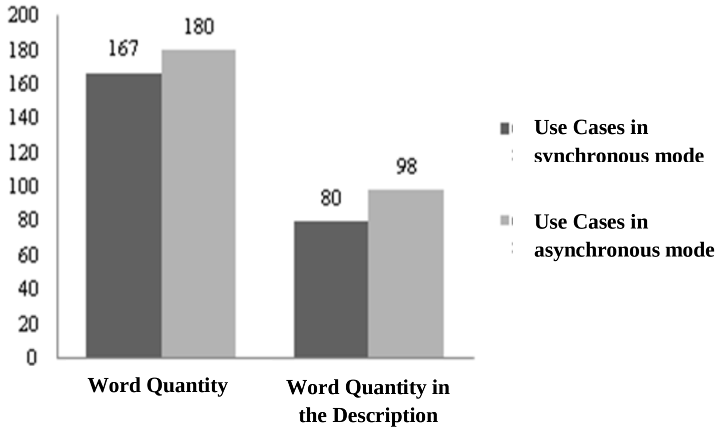


Figure 11: Word average quantity in Use Cases

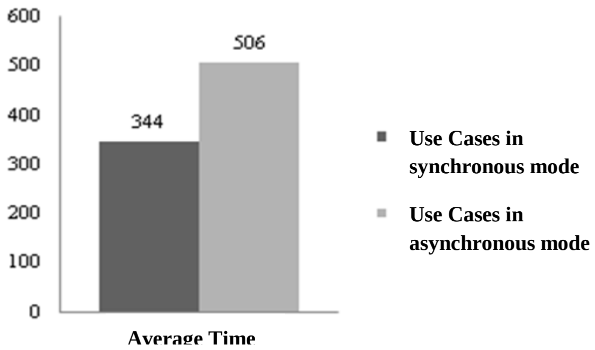


Figure 12: Average time used to generate Use Cases

5. CONCLUSIONS AND FUTURE WORK

Results help to obtain more information, which conducts to validate the hypothesis that asynchronous mode facilitates the redaction of better Use Cases, because the cases produced in this mode reach better satisfaction from the client and experts in Use Cases creation and redaction, have better text structure, less implementation slants, fits better in a page and have less phrases and words that would make the case ambiguous, i.e., they have better quality. On the other hand, they take longer time to produce cases, taking an average of 1.5 times hours-man than the synchronous mode.

Also, a set of metrics was obtained, which can be used to evaluate the Use Cases generated by people, using a collaborative tool or not, in future research work or even developing new evaluation tools for Use Cases. On the other hand, the metrics proposed in this investigation can be modified or some new metrics can be generated, and doing future research to help find a way to work for a group of people, through a collaborative tool to produce software specifications.

It came to the light, also, by each of these experiences, that while working with people it is important to take care to adapt to their schedules and interact with them, so empathy and understanding about their feelings or thoughts is mandatory.

A future investigation could use groups of people working synchronously for a longer time (giving them a minimum of time to work, which must be respected); with some kind of resting time so they can perform well until the end of the experience, or using people working with the synchronous collaborative tool in the same room. These results can be compared with the results of groups of people using an asynchronous collaborative tool for similar time frames. It is also proposed to have experiences in a work environment, with a real project, aiming to compare those results with the ones described in this document.

For future work it is considered to improve the existent collaborative tools, advising to increase their usability, improving their graphic interface, adding more communication means between participants (videoconference, interactive chat, voice calls, etc.). It is advised to allow a greater customization of the writing for each participant in the collaborative editor (choosing font, color and size for each participant), to have a way to know who is writing in a determined moment and that the users can edit collaboratively the other fields used to write Use Cases, not only the description.

The creation of new collaborative tools for Use Cases production, asynchronously-orientated, could also be guided, because this mode helps to produce better Use Cases.

References

[1] INCOSE. *Requirements Management Tools Survey*. Recovery: 09 October 2008, from: http://www.paper-review.com/tools/rms/read.php.

[2] Ramírez, C., *Sistema colaborativo para captura de requisitos de software mediante Casos de Uso*. Trabajo de Título de Ingeniería Civil Informática. Santiago: Universidad de Santiago de Chile, 2002.

[3] Cano, E., *Sincronización lógica en el espacio de trabajo compartido en sistemas colaborativos*. Trabajo de Título de Ingeniería Civil en Informática. Santiago: Universidad de Santiago de Chile, 2000.

[4] Sanhueza, R., *Rediseño de una herramienta colaborativa de captura de requisitos basada en las recomendaciones de una evaluación de usabilidad*. Trabajo de Título de Ingeniería Civil Informática. Santiago: Universidad de Santiago de Chile, 2005.

[5] Pulgar, J., *Integración de una herramienta colaborativa sincrónica y otra Asincrónica para el apoyo a la construcción de Casos de Uso*. Trabajo de Título de Ingeniería Civil Informática. Santiago: Universidad de Santiago de Chile, 2006.

[6] Antillanca, H., Fuller D., "Sistemas colaborativos sincrónicos cara-a-cara: requisitos, problemas y resultados". 3er. Encuentro Chileno de Computación, 1995.

[7] Whitehead, J., "Collaboration in Software Engineering: A Roadmap". *International Conference on Software Engineering 2007: Future of Software Engineering*, vol. 1, pp. 214-225, 2007.

[8] Sommerville, I., *Ingeniería de Software*, 7th edition, editorial Pearson, Madrid, 2005.

[9] DeSanctis G., Gallupe B., "A foundation for the study of group decision support systems". *Management Science*, vol. 33, No. 5, (May 1987), pp. 589-609.

[10] Jaaksi, A., "Our Cases with Use Cases". *JOOP (Journal of Object-Oriented Programming)*, vol. 10, No. 9, (February 1998), pp. 58-65.

[11] Jaaksi, A., Aalto, J., Aalto, A., Vättö, K., Tried and True Object Development: Industry-Proven Approaches with UML, Cambridge University Press in association with SIGS Books, UK, 1999.

[12] Techsmith. *Camstasia Studio, Screen Recorder Software*. Recovery: 01 September 2008, from: http://www.techsmith.com/camtasia.asp

[13] Hernández, R., Fernández, C., Baptista, P., *Metodología de la Investigación*, 3th edition, editorial McGraw-Hill/interamericana Editores, Chile, 2003.

[14] Wilson, W., Rosenberg, L., Hyatt, L., “Automated Analysis of Requirements Specifications”. *International Conference on Software Engineering, 19th international conference on Software engineering*, vol. 1, pp. 161-171, 1997.