APPLYING ROLE-PLAYING GAME IN SOFTWARE DEVELOPMENT SUBJECTS

S. Rueda, J.I. Ignacio Panach, J.B. Cabotà, F. Valverde

Departament d'Informàtica, ETSE-UV (SPAIN)

Abstract

Current trends on education are focused on introducing practical activities as similar as possible to real problems that students will face within their professional works. Curriculums in universities have several subjects related to good practices on software development (UML models, development methodologies, design patterns, testing, etc.). However, in all of them, the software to develop is usually specified through text. This contrasts with what happens in the labor market, where usually developers must understand the software requirements through interviews with the end users.

In order to involve students in a context similar to professional life, we have applied a role-playing game where they elicit the software requirements through an interview with the teacher that plays the role of the end user. This approach has several advantages compared to providing a textual description of the system to develop. Using role-playing, students develop a set of skills that they do not develop if the system specification is presented textually. Next, we highlight three main advantages. (1) Students must learn a technique to elicit requirements through interviews. They must learn, what to ask, what to report and how to conduct the interview. (2) Students have to develop the aptitude of working in a team of developers, since the whole development is performed in groups of 5 or 6 students. This forces the students to report all the data extracted from the interviews in order to share all the knowledge among all the members of the team. (3) Students have to learn how to validate their ideas with end users. Finally, the teacher, who plays the role of end user, must check whether or not the requirements understood by the students are the same as needed.

The approach was applied in the subject of Software Engineering at the Escuela Técnica Superior de Ingeniería (ETSE) in the Universitat de València, Spain, within two degrees: Telematics Engineering and Computer Engineering. Every year since 2010, students must develop a software project from Requirements specification to Java Code using UML specifications. Previous years, software specification was shared with students through a textual description. The last year, we have proposed a project with similar difficulty than previous years, but using the role-playing game approach to avoid the textual description. In order to compare requirements specification quality and student engagement using a textual description versus interviews with role-playing game, we have applied each treatment to each degree. Students of Telematics Engineering had the textual description of the system specification while students of the degree in Computer Engineering had to interview the teacher who played the role of the end user. This way, we can study pros and cons of role-playing game compared with a textual description in the development of the same project.

From the analysis of results, we extract several conclusions. The application of role-playing game approach is similar than using textual approach in terms of quality of software requirements specification. The time to assist to the interviews, to report the extracted data, to coordinate the interviews and to validate with the end user is an extra time that only took students with roleplaying game. Although the effort of the students is higher in the role-playing game, their motivation and engagement in the project is better than with a textual description.

Keywords: Innovation, role-playing game, software development projects, requirements elicitation.

1 INTRODUCTION

In general, subjects related to software engineering are based on developing a software system from scratch. Usually, the description of the system to develop is shared among the students textually. So students only have to read all the requirements that appear in the text. This way of eliciting requirements is quite conformable for both students and teachers. Teachers know what to demand to students and students know what they must implement. However, textual definition of requirements is far from a real development context, where usually the client does not know exactly what are the requirements [6] and the developer must guide a set of interviews to elicit those requirements.

This paper tackles our experience of trying to submerge the student in a context as real as possible through playing a role game [7]. The students play the role of developers and the teachers play the role of client. So, student must arrange a set of interviews with the teachers to elicit the requirements. This is pilot approach that we have conducted during the last course. In previous courses, the requirements were shared textually among students.

The approach was applied in the subject of Software Engineering at the Escuela Técnica Superior de Ingeniería (ETSE) in the Universitat de València [2], Spain. In this subject, students have to develop a software system from scratch. First they perform the analysis with Use Cases and Class Diagrams, next they perform the Design with Sequence Diagrams and finally they must implement the code with Java. We conducted the experience in two degrees: Telematics Engineering and Computer Engineering. Students of Telematics Engineering is our control group, we have applied the same treatment than previous years, we shared the description of the requirements textually. On the contrary, the students of Computer Engineering had to elicit the requirements through interviews with the teachers (new treatments). This way we can compare both treatments to study pros and cons of role playing game.

Results of the experience show that the quality of the developed product is similar with both treatments. However, the time spent with the role playing game is higher. Students have to prepare the interviews, assist to the meetings and report all the requirements elicited after each meeting. Note that all these tasks were not necessary for students with a textual description. However, we notice that motivation of students with role playing game is considerably better. They are engaged with the project from the early sessions, while students with a textual description take more time to show interest in the development.

Next we summarize the sections of the paper. Section 2 describes the teaching process. Section 3 explains the results and discusses ideas from them. Finally, section 4 shows the conclusions of the work and possible future research lines.

2 TEACHING PROCESS

This section describes the process followed in Telematics Engineering and in Computer Engineering. Fig. 1 shows the process conducted in the degree of Telematics Engineering, where teachers share a textual description of the system to develop. This control treatment is the same process applied in the degree during previous courses. Next we describe each step:

1 The teachers build groups between 5 or 6 students.

2 The teachers make public a textual description of the problem description. All the groups develop the same problem.

3 After 15 days, the students have to draw a use UML [1] Case Diagram together with its requirements templates. Moreover, a textual description of functional and non-functional requirements is also submitted. As summary of results, the students must also submit a spreadsheet with some numerical data that shows at a glance a summary of the whole process of eliciting and documenting the requirements from the textual description of the problem.

The Use Case diagram and requirements templates are the input for the next stages of the software development process: analysis, design and implementation.

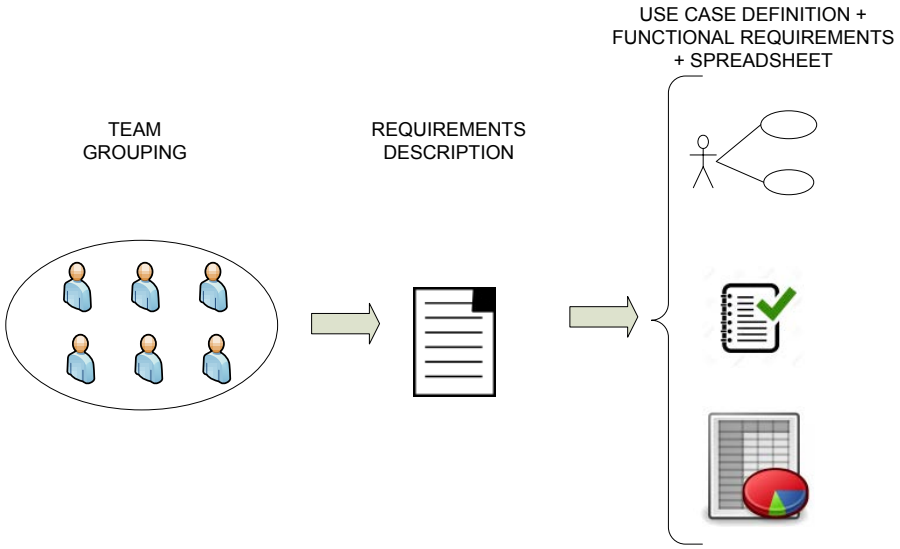

Fig. 1. Process for control treatment (textual description of the requirements)

The new treatment applied to the degree in Computer Engineering makes use of the role playing game. In this case, students must elicit requirements through interviews with the teacher. Fig. 2 shows the process from scratch, next we explain each step.

1 The teachers build groups between 5 or 6 students (the same as in the control treatment)

2 The students have to prepare the interview. For this aim, the teachers provide material to study several techniques for the interview: prototypes [3], JAD [5], unstructured interviews, etc. This preparation is done out of the classroom as homework. The problem to solve is the same for all the teams.

3 Each team arranges a meeting with the teacher individually. During the meeting, the teacher answers the questions of the students playing the role of client. Each team applies one of the technique suggested by the teacher to guide the interviews. The teacher does not correct in case the technique is not applied properly (the teacher only plays the role of client). In case the students have any question regarding the technique to guide the interviews, they can be answered in another meeting with the teacher to deal that topic.

4 Once the students leave the meeting, they must report all the elicited requirements. If the students think that there are some requirements that are confusing or missing, they can arrange again another meeting. In general, each team arranged around 3 meetings.

5 After 15 days, the students draw a UML Use Case diagram and its templates in the same way as in the control treatment. Moreover, a textual description of functional and non-functional requirements is also submitted. As summary of results, the students must also submit a spreadsheet with some numerical data that shows at a glance a summary of the whole process of eliciting and documenting the requirements throughout the interviews.

The Use Case Diagram is the input for the next development stages. Next stages are identical for both treatments.

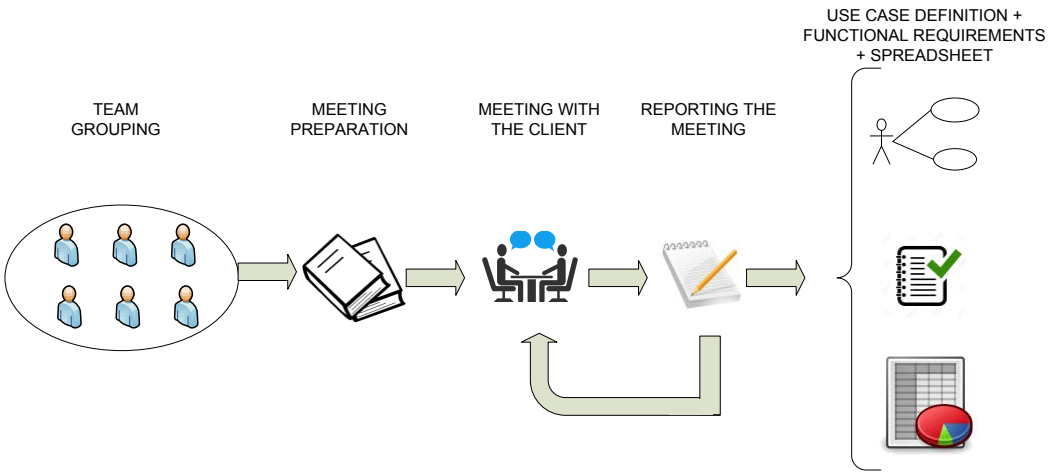

Fig. 2. Process for new treatment (requirements elicitation through interviews)

2.1 Instruments

This section describes the instruments that students used to conduct the experience. There are some instruments that are the same for both treatments and other that are treatment-specific. Instruments used in both treatments are:

- **A problem to develop**. The problem is a system to manage a hotel. Most frequent actions in this system are: look for room availability for a given date; check in; check out; cancel reservation; manage customers, etc.
- **A tool to draw UML models**. We work with Visual Paradigm, since it supports all the UML diagrams. Students has access to an academic license to work from their house.
- **A SVN repository**. A collaborative repository is useful to share the UML diagrams
- **A spreadsheet**. This document reports numerical data of the whole process to build the Use Case diagram and the requirements document from scratch. The elaboration of this spreadsheet is based on the work of Dieste et al. [4]. The spreadsheet is the same for both treatments except for the two last metrics of performance, which are exclusive for the role playing treatment. Table 1 shows the data asked in the spreadsheet. Quantitative data extracted from this spreadsheet is useful to compare pros and cons of both treatments

Table 1. Data provided through a spreadsheet to report the process from scratch.

Quantity	Ratio of functional requirements
	Ratio of non-functional requirements
	Ratio of overall requirements
	Ratio of pages used in the document
Diversity	Ratio of overlapped requirements
Completeness	Ratio of professor matched requirements
Quality	Ratio of new requirements
	Ratio of bad specified requirements

Instruments that are specific for the treatment with role playing game are:

- **A set of electronic books.** The teacher makes available some electronic books to learn how to apply the techniques to guide the interviews. Students must learn these techniques on their own but the teacher could answer any question in the classroom (not during the interview where they are applying the role of client).
- **A word processing tool**. It is used to write down the reports with the results of the meetings.

3 TEXTUAL DESCRIPTION VERSUS ROLE PLAYING GAME: RESULTS

As we have mention before, we have measure different parameters in order to compare both subjects. We have divided these parameters in four categories. First category is Quantity, that includes four parameters related to the amount of requirements students have included in the specification document. As we can see in figure Fig. 3, we have measured the number of functional requirements, the number of non-functional requirements, the total number of requirements (functional and non-functional) and the amount of pages used to describe them. For comparison porpoise, we show the ratio between the student's number and the professor's number. The control group (Telematics Engineering) is labelled as GIT and the other group (Computer Engineering) is labelled as GII. As we can see, the control group provided more requirements (either functional or non-functional) and more pages. However, as we will show later, it doesn't mean that all of them were correct. In fact, is completely normal that students in the group without statement will find it difficult to obtain requirements.

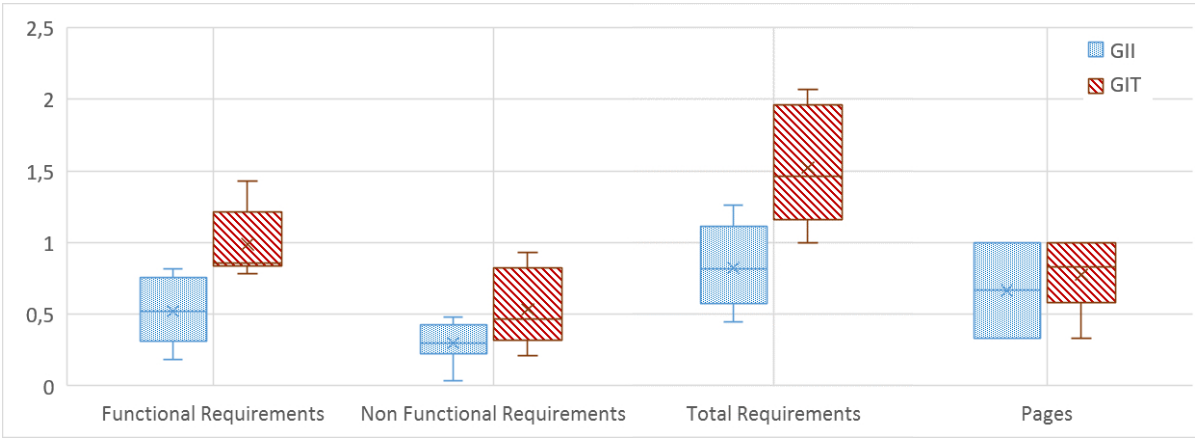

Fig. 3: Quantity

Second category is Diversity, that includes just one parameter related to the amount of overlapped requirements that students have included in the specification. That is the number of requirements (functional or non-functional) that in fact are describing the same thing, so we cannot consider them as different requirements. Figure

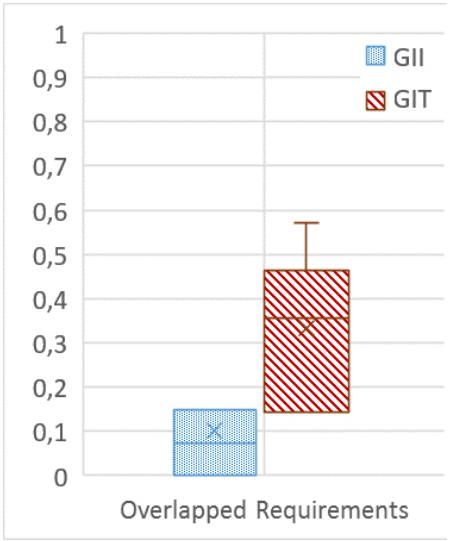

Fig. 4: Diversity

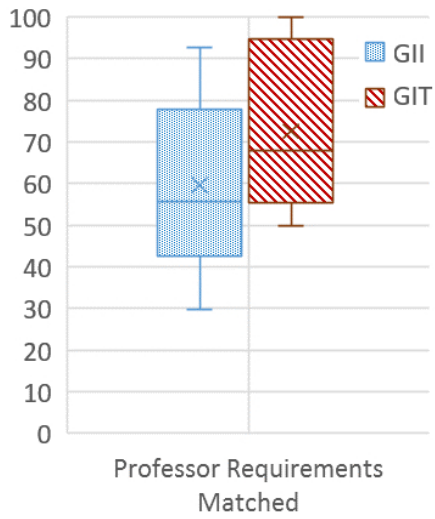

Fig. 5: Completeness

shows the ratio of overlapped requirements and the total amount of requirements proposed by professor. In that case, the control group also presents a higher ratio of overlapped requirements. We think this is because these students had worked less in the elicitation process.

Completeness is the third category of parameters studied. In that case, we have measured if functional and non-functional requirements in the students' solution that matched with any requirement in the professor solution. In that point, we have no taken into account if it was right described nor qualified as a functional or non-functional requirement, just if it was similar enough to any professor requirement. In Figure Fig. 5 we can see the results for the percentage of requirements included in the students solutions that matched a requirement. As we can see, again this time the control group obtained better results. However, this was expected as they were provided with the statement.

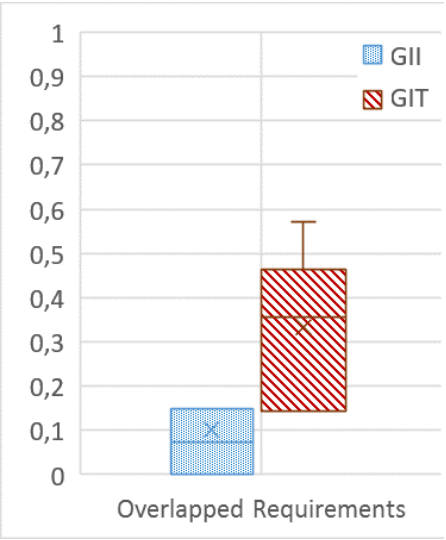

Fig. 4: Diversity

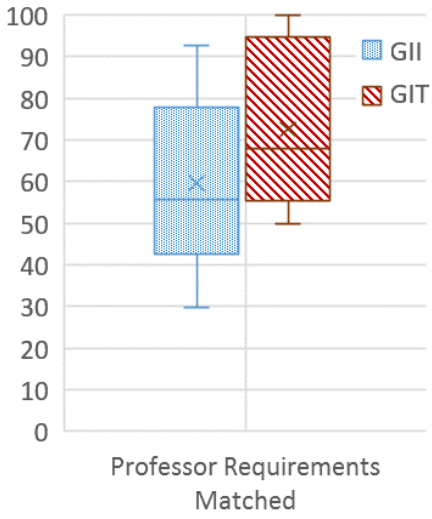

Fig. 5: Completeness

Finally, the last category of parameters studied is Quality. We have included here two different parameters. First, New Requirements, represents the requirements included by students that are not present in the professor solution, regardless of type nor if they are properly described. Then, Bad Specified Requirements, denotes the amount of requirements that students did not describe correctly, they were not well classified or that they had nothing to do with the application that was asked. We think this is the most significant parameter in order to determine the impact that the proposed approach has in the ability of students to perform correctly an elicitation process.

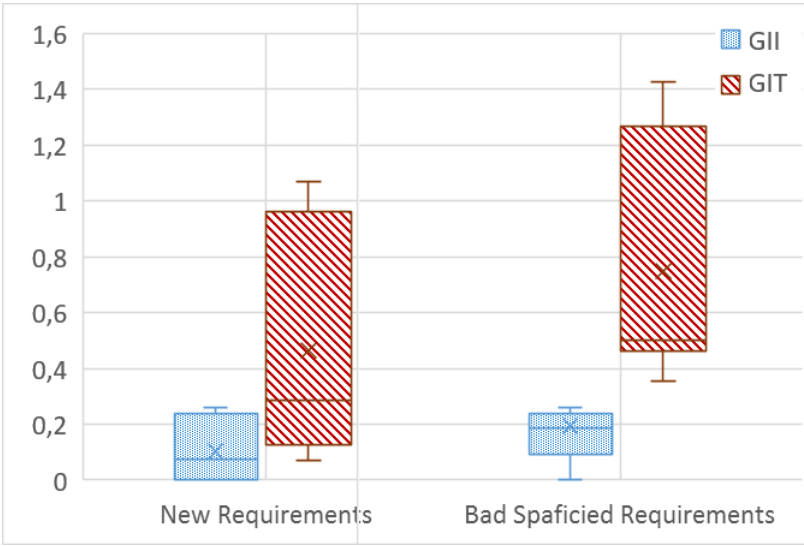

Fig. 6: Quality

In Figure Fig. 6, we have represented the ratio of new requirements and the number of professor requirements. As we can see, in that the control group provided higher ratios of both parameters. In the case of new parameters, in a few cases, students work groups discovered some valid new requirements that professor had not had in mind and that were acceptable (which it does not mean

they were properly described). Conversely, in many other case, the new requirements had nothing to do with the project propose. That was more common in the control group.

Moreover, in Figure Fig. 6, we have represented the ratio of bad specified requirements and the number of professor requirements, In that case, we can see that the control group presents worst results, showing that although requirements were obtained, they were not specified correctly. We think that this is due to less involvement of students in control group.

4 CONCLUSIONS

In this paper, we have presented our work carried out at the Software Engineering Department at de University of Valencia in order to improve the teaching methodology of the requirements elicitation process in Software Engineering subjects with a role playing game. Results show that with the role playing game, students obtains less quantity of requirements. However, they provide better quality results in terms of correctness specification of requirements. We think this is due to the more quantity of time these students spent on the process. Moreover, we also have detected that these students were more motivated that the ones in the control group.

ACKNOWLEDGEMENTS

This work was promoted at the Universitat de Valencia under the project “Desarrollo de software desde una perspectiva industrial” (UV-SFPIE_GER16-418809- Grupos estables y redes de innovación continua)

REFERENCES

[1] *UML: http://www.uml.org/*.

[2] Universidad de Valencia: *http://www.uv.es*.

[3] A. Cohen, *Prototype to Product*: O'Reilly Media, Inc., 2015.

[4] O. Dieste and N. Juristo, "Systematic review and aggregation of empirical studies on elicitation techniques," *IEEE Transactions on Software Engineering,* vol. 37, pp. 283-304, 2011.

[5] E. G. Mallach, *Information Systems*: CRC Press, 2015.

[6] K. Wiegers and J. Beatty, *Software requirements*, 2013.

[7] D. Zowghi and S. Paryani, "Teaching requirements engineering through role playing," in *Proceedings of the 11th IEEE Joint International Requirements Engineering Conference (RE03)*, Monterey Bay, CA. 2003