Requirements Elicitation Methods based on Interviews in Comparison: A Family of Experiments

Silvia Rueda[1], Jose Ignacio Panach[1], Damiano Distante[2]

[1]Escola Tècnica Superior d'Enginyeria, Departament d'Informàtica, Universitat de València
Avenida de la Universidad, s/n, 46100 Burjassot, Valencia, Spain
{joigpana, silvia.rueda}@uv.es

[2]University of Rome Unitelma Sapienza
Viale Regina Elena, 295, 00161 Rome, Italy
damiano.distante@unitelmasapienza.it

Abstract. Context: There are several methods to elicit requirements through interviews between an end-user and a team of software developers. The choice of the best method in this context is usually on subjective developers' preferences instead of objective reasons. There is a lack of empirical evaluations of methods to elicit requirements that help developers to choose the most suitable one. **Objective:** This paper designs and conducts a family of experiments to compare three methods to elicit requirements: Unstructured Interviews, where there is no specific protocol or artifacts; Joint Application Design (JAD), where each member of the development team has a specific role; Paper Prototyping, where developers contrast the requirements with the end-user through prototypes. **Method:** The experiment is a between-subjects design with next response variables: number of requirements, time, diversity, completeness, quality and performance. The experiment consists of a maximum of 4 rounds of interviews between students that play the role of developers and an instructor that plays the role of client. Subjects had to elaborate a requirements specification document as results of the interviews. We recruited 167 subjects in 4 replications in 3 years. Subjects were gathered in development teams of 6 developers at most, and each team was an experimental unit. **Results:** We found some significant differences. Paper Prototyping yields the best results to elicit as many requirements as possible, JAD requires the highest time to report the requirements and the least overlapping, and Unstructured Interviews yields the highest overlapping and the lowest time to report the requirements. **Conclusions:** Paper Prototyping is the most suitable for eliciting functional requirements, JAD is the most suitable for non-functional requirements and to avoid overlapping, Unstructured Interviews is the fastest but with poor quality in the results.

Keywords: Requirements elicitation, empirical software engineering, prototyping, joint application design

1 Introduction

A requirement is *a condition or capability that must be met or possessed by a system or system component to satisfy a contract, standard, specification, or other formally imposed documents* [1]. There are several methods to elicit requirements, including interviews, questionnaires, task analysis, domain analysis, introspection, card sorting, among others. Interview-based elicitation methods are probably the most traditional and commonly used method for requirements elicitation [2, 3]. How the interviews are conducted, what to do within the interviews, how to report the interviews and how to create artifacts from these interviews is left to the developers' choice. As a solution to guide the interviews, several methods have been proposed. All these methods aim at involving the client or end-user in the design and development process of the application starting from the early steps of requirements elicitation phase.

This paper reports on an empirical study conducted to compare, in practice, three different requirement elicitation methods based on interviews, namely Joint Application Design (JAD), Paper Prototyping, and Unstructured Interviews, and analyze the pros and cons of each method working in groups of developers. JAD [4] is a method conducted in a team where each team member plays a different role with specific tasks in the interview. This results in well-structured interviews. Paper Prototyping is a method where the group of developers build paper prototypes to describe the elicited requirements, gather detailed information and offer immediate feedback [5]. Unstructured Interviews are interviews in which developers of the team do not have specific roles, nor there are concrete steps to follow or preconfigured questions to use during the interview.

The choice of these three methods was motivated by their high use in software development methods and their diversity. JAD is a strongly structured method; however, the end-user is not directly involved in the application development process beyond answering questions to elicit requirements. Paper Prototyping is less structured; if applied in a team, each analyst does not have specific tasks to perform, and more complex tasks can be shared among different analysts. It also allows involving the end-user from the first steps of the development process, since the end-user can participate in the elaboration and validation of prototypes. Finally, Unstructured Interviews is the least structured

method; similarly, to Paper Prototyping, when applied in a team, each analyst can perform the task that she/he considers more suitable. In this case, the end-user is not involved in the process beyond answering questions to elicit requirements (like JAD). According to previous publications in the literature, JAD is the least used in practice among our three considered methods [6] while the most frequently used is Unstructured Interviews [7] in which each analyst conducts the interview her/his best.

The main contribution of this paper is the design and the conduction of a family of experiments to compare JAD, Paper Prototyping and Unstructured Interviews as requirement elicitation methods. The family of experiments has been replicated during three years with undergraduate students of the course on Software Engineering given in the degrees in Computer Engineering and Telematics Engineering, with two replications in each degree (4 replications in total). We recruited 167 subjects adding all replications. As Basili [8] states, the replication of the experiment contributes to building a body of knowledge by combining and generalizing results. In the three elicitation methods studied, the same instructor played the role of the client. Students, gathered in groups of 6 people maximum, had to elicit requirements through interviews with the client. Each group of students only applied one elicitation method, and all of them had to report the results of the elicitation using a predefined format based on a simplified version of the IEEE 830-1998 standard [9].

Each team of developers is an experimental unit in the experiment. Dieste et al. [10] conducted a systematic review of empirical studies on requirements elicitation techniques. The comparison reported in that review was done through a set of response variables that measure the characteristics of each method. We use the same variables as response variables in our study: *number of requirements* (functional and non-functional), *time* (to prepare the interview, within the interview, to report requirements), *diversity* (overlapping among requirements), *completeness* (percentage of requirements regarding the instructors' solution that were identified), *quality* (percentage of requirements that appear in the instructors' solution minus the percentage of requirements that are not part of the instructors' solution), and *performance* (number of questions asked in the interview and efficiency, i.e., the ratio of quality per time). The empirical study is based on a between-subjects design, where each subject is assigned only one elicitation requirements method.

As significant results, we found that Paper Prototyping is the method that, compared to JAD and Unstructured Interviews, identifies the highest number of requirements; JAD is the slowest method to write down the requirements and the method with the best diversity (the lowest overlapping); Unstructured Interviews is the fastest method, the method that elicits fewer requirements and the method with more overlapping. Apart from significant differences, we found that Paper Prototyping is the method with the best completeness, the best quality, and the best performance. JAD is the best method to elicit non-functional requirements and the method that asks more relevant questions. Unstructured Interviews is the method with the least time spent to prepare interviews, time spent during the interviews and time spent to report the requirements document. As negative characteristics, Paper Prototyping is the worst method to elicit non-functional requirements. JAD is the method that requires more time within the interviews and the method with less efficiency in the elicitation process. Unstructured Interviews is the method that found the least number of functional requirements, the method with the least number of questions asked and the method with the lowest quality.

This paper is structured as follows. Section 2 describes other related works that conducted experiments in the requirements elicitation area. Section 3 introduces the tasks assigned to the students and the characteristics of the three elicitation methods analyzed in the experiment. Section 4 shows the design of the experiment. Section 5 deals with the results of the quantitative data through statistical analysis and descriptive results. Section 6 presents the discussion of the results, highlighting the pros and cons of each method. Finally, Section 7 shows the conclusions and future works.

2 Related Work

In general, requirements are classified as functional and non-functional, and elicitation methods behave differently depending on what type of requirement to elicit. For example, Sharma and Dhir [11] have analyzed the differences of both types of requirements from the point of view of the methods for their elicitation. There are several works that have analyzed the different requirements elicitation methods for both functional and non-functional requirements. Proof of the existence of many elicitation methods is the systematic literature review conducted by Rodriguez et al. [12]. This review analyzes what aspects can be covered with requirements elicitation, the different activities needed during the elicitation process, and what factors have an influence on requirements elicitation. Results show that there are still important challenges to improve the requirements elicitation methods. Some of these challenges are related to ambiguities during the whole process. Other works such as the work of Carrizo et al. [13] analyze what is the best requirements elicitation method depending on the context. A framework to help requirements engineers select the most adequate method was defined and validated.

Next, we describe works that deal with the **validation of requirements elicitation methods based on interviews**. Spoletini et al. [14] proposed a protocol, empirically validated, to conduct reviews of requirements elicitation interviews. The protocol is based on recording the interviews such a way these recordings are inspected by both the analyst who

performed the interview and another reviewer. Ferrari et al. [15] proposed to investigate the differences between ambiguities explicitly revealed by an analyst during a requirements elicitation interview and ambiguities annotated by a reviewer who listens to the interview recording. Results show that ambiguities revealed by the analyst do not match with the ambiguities found by the reviewers. This highlights the impact that ambiguities can have in the requirements elicitation process. Another challenge is related to the trust in requirements elicitation. Burnay and Snoeck [16] conducted an empirical study to measure the impact of trust during requirements elicitation, differentiating the role of the developer and the client. An important challenge of requirements elicitation is the requirements validation and verification area, which, despite its recognized importance, lacks empirical research. On this regard, Ambreen et al. [17] highlight the necessity to replicate studies of requirements elicitation. The target of our paper is aligned with this challenge: the replications of empirical experiments in the area of requirements elicitation.

Some authors have analyzed **existing issues in the process of requirements elicitation**, such as Kumari and Pillai [18] [19]. These authors have proposed a contingency model to deal with issues during the requirements elicitation. The model aims to recommend steps during the whole elicitation process. Farinha and Mira da Silva [20] deal with the analysis of issues in the focus groups method. They conducted a study that outlined some issues of this method: stakeholders' difficulties recognizing and articulating their own needs; conflicts of stakeholders' interests and analysts' misinterpretations.

There are previous works that have **empirically analyzed several requirements elicitation methods**. Besrour et al. [21] have conducted an empirical study to assess and evaluate JAD, Interview, and Brainstorming. Results show that there are differences among them, concluding that Interview is the method with the best results. Brill et al. [22] have compared use cases versus videos to know which method elicits more requirements. Results show that even low-effort videos can work better than use cases for avoiding misunderstandings in the early phases of a development project. Nascimento et al. [23] compared two methods to specify use cases: textually and graphically. The results show that textual form and graphical-based specifications present similar levels of correctness and that the time spent to create them is also similar. Ibriwesk et al. [24] focus their study on four requirements elicitation methods based on textual documents: Natural Language, Data Flow Diagram, Perspective-Based Reading, and Archeology-Based Reading. Results show that Natural Language yields the best results in quality, effectiveness, understandability, facility, and number of difficulties. Hadar et al. [25] conducted an empirical study to analyze the perceived and actual effects of prior domain knowledge. Results indicate that domain knowledge affects elicitation via interviews in communication with the clients and understanding of their needs. Lloyd et al. [26] evaluated the effectiveness of elicitation requirements in distributed requirements. Results indicate that the active participation of the client in the process is essential to get an effective list of requirements. As artifact to report requirements, the study concludes that a textual requirements document is the best one. Kumari and Pillai [18] conducted an empirical study of a contingency model to depict the influence of requirements elicitation issues on project performance. Results show that recommendations provided by the contingency model help to elicit requirements effectively. Mahmood and Ajila [27] analyzed the impact of multi-natural language backgrounds of the developers on the quality and correctness of the case modeling. This study was conducted through a controlled experiment. Results show that using a native language for system description improves the functional correctness of the use case model. However, the time required to define the use case and its quality are not affected by the native language.

Some of the empirical studies consist of a **family of experiments including several replications**. Aranda et al. [28] conducted a replicated controlled experiment to study the influence of the analyst's problem domain knowledge on elicitation effectiveness. Results show that analyst's problem domain knowledge has a small effect on the effectiveness of the requirements elicitation process. The interview has a significant positive influence, as does general training in the requirements elicitation method. Riaz et al. [29] conducted three replications to study the use of security requirements templates. Even though the target of the family of experiments was security, the relationship between security and functional features was also considered. Results yield that subjects who used the proposed templates identified easily relevant suggestions in the elicitation process. Potential limitations on knowledge in security may affect the use of the templates negatively.

Other works have **proposed methods to elicit requirements** and have validated them with subjects. Morales-Ramirez [30] proposed a linguistic method based on speech-acts for the analysis of online discussions. The goal is to discover requirements-relevant information from such discussions. Results of the validation yield evidence that there is an association between types of speech-acts and categories of issues, and that there is a correlation between some of the speech-acts and issue priority. Adam and Schmid [31] proposed a problem-oriented method for software product lines (SPL). The idea was to reuse requirements for the development of different systems. Results of the validation yield that the use of the approach is more effective than a traditional method. Gralha [32] proposed a method to evaluate requirements models through eye-tracking. The idea is to collect metrics about the model during the modeling process and the subjective opinion of the clients. Gacitua et al. [33] defined a method for the identification of abstract requirements through ontology learning. The proposal aims to assist the developer during the requirements elicitation process. Results of the validation show that human judgment is still needed in abstract identification of requirements. There is no way to identify all abstract requirements automatically. Vitharana et al. [34] defined mental models for improving the require-

ments elicitation process. The mental model considers the different aspects that may affect the quality of the requirements: the tool to support the process and the multiplicity of perspectives. The approach includes measures for analysts' mental models and requirement performance. Results support the mental model and show a high impact of the tool that helps the elicitation process. Yamanaka and Komiya [35] have proposed a method based on interviews to navigate among requirements. The goal is to elicit as many requirements as possible without omissions or errors. Results of the validation of the proposal show that requirements elicitation work depends on the business knowledge and work experience of each subject, resulting in better efficiency of requirements elicitation. Browne and Rogich [36] proposed a model of the requirements elicitation process to construct a new requirements elicitation prompting technique. The approach was evaluated in terms of effectiveness comparing it versus two other techniques (interrogatories and semantic questioning scheme). Results show that the proposal elicited more quantity of requirements.

The idea of **using games to facilitate the requirements elicitation** has also been tackled in several works. Ghanbari et al. [37] proposed a method based on online serious games for gathering requirements from distributed software stakeholders. Results of the validation yield that the approach was pleasant and easy for subjects with less technical experience. Gamification tends to enhance innovation and creativity among end-users and also facilitates collaboration and communication among stakeholders. Lombriser et al. [38] developed a gamified requirements model to deal with requirements elicitation. The aim is to get client engagement to improve the quality of the requirements. Results of applying the proposal show that the performance of the subjects is higher and that the defined requirements are more numerous, with higher quality and more creativity.

As a conclusion of the related works, we highlight that there are many methods to elicit requirements. More importantly, even though there are many empirical validations of requirements elicitation methods, only a few works ([21],[22],[23],[24]) focus the analysis on the comparison among different ones. This implies that there is a lack of empirical studies on the pros and cons of existing methods. This lack is even wider when we look for a family of experiments. We only found two families of experiments ([28],[29]) in all the related works and none of these families deal with the comparison of requirements elicitation methods. As Basili [8] states, replications contribute to building a body of knowledge combining and generalizing results. To help filling the gap in families of experiments that compare different requirements elicitation methods, this paper describes the design and the results of a family of experiments that compare three requirements elicitation methods based on interviews: Unstructured Interviews, JAD, and Paper Prototyping. The choice of methods based on interviews was justified by their broad use in the practice. On this regard, most of the methods analyzed in the related works were based on interviews. The high frequency of use and diversity of the three methods considered motivated their choice among other methods based on interviews.

3 Studied Requirements Elicitation Methods

The goal of our study is to analyze and compare three requirements elicitation methods based on interviews widely used in the software engineering field: Unstructured Interviews, JAD, and Paper Prototyping. In our experiments, we assigned one method per team, and the team had to learn the method on their own using manuals elaborated by the instructors. For each method, subjects could arrange from two to four interviews (they usually took three) with the instructor that played the role of the client. After the interviews, subjects had to elaborate a requirements specification document that specifies all the elicited requirements. This document was the artifact analyzed in our experiment. Next, we describe each method in detail, with the characteristics of each one.

3.1 Unstructured Interviews

This method consists in eliciting requirements through interviews with the client but without following a specific protocol or producing specific artifacts. We offered a manual with some recommendations about what type of questions they could ask and how to proceed in the interview. This manual also encouraged subjects to prepare a list of questions considering the following tips extracted from previous works in the literature [39] (chapter 12, Interviews and Questionnaires):

- The list of questions should be organized into a logical order, starting from the most general to the most specific, and arranged as groups of questions about related issues.
- The team must decide how much time to devote to each issue.
- It is quite difficult to prepare all the questions in advance; we recommend using the information got during the interview to formulate additional questions.

We described how to conduct the interview [39] (chapter 12):

- Introduce the members of the team.
- Review the goals of the interview.
- Improve the understanding by summarizing, rephrasing, showing implications.
- Be an active listener.
- Be courteous.

- Use non-verbal communication methods.
- Use open-ended questions to encourage unconstrained answers.
- Use close-ended questions to force a precise or detailed answer.
- Do not anticipate the answers.
- Ask questions that approach the issue from different directions.
- Ask the questions to raise the level when the interview begins to get too detailed or too focused.
- Check for errors periodically, recognize when they occur, and correct them.

The process should be the same in all interviews with the client. Any member of the team can ask questions, the client answers them and anyone takes notes of the responses. There is no other artifact that the notes written in the interviews. The first meeting is mainly used for listening to the client and the other ones are more focused on asking specific questions on fuzzy requirements. At the end of this process, the team elaborates the requirements specification document with the notes taken from the meetings.

3.2 Joint Application Development (JAD)

JAD consists in dividing the members of the team into different roles, where each member has specific tasks to elicit the requirements [4]. This method also aims to collect requirements through interviews with the client. In addition, by assigning specific roles to team members, it aims to increase their interest and, consequently, the quality of specifications.

JAD is divided into three phases:

- Customization: The leader coordinates the members of the team.
- Session: The team interviews the client to elicit requirements.
- Wrap-up: The team writes down the requirements elicitation document.

JAD works with the following roles:

- Session leader: The leader must mediate political disputes, power struggles, and personality clashes. The leader needs to be impartial and able to keep an open mind and control controversies.
- Executive sponsor: This person bears the financial responsibility for the system. This person must decide whether the project goes on or not.
- Recorder/Scribe: This is someone whose primary responsibility is to record what happens during the session. The scribe is an active participant that asks for clarification and points out inconsistencies.
- Developers: They must learn the system during the interviews with the client. Their role is to provide information, not to make judgments of any user ideas.
- Experts/Client: This is the person that knows the requirements and requests the development of the software system.

We provided each team with a manual with the different phases of JAD, the different team members and their roles. The distribution of roles among the members of the team was done according to their preferences without the intervention of the instructors. Roles were permanent during the whole experiment. The process is the same through all the interviews. As in Unstructured Interviews, the first interview is usually more oriented to listen to the client and the others are more oriented to ask more details about questions that are not clear from the previous interviews.

3.3 Paper Prototyping

Paper Prototyping is a method to draw a paper prototype with a visual representation of what the system will look like [40]. It can be hand drawn on a paper or created by using a graphics program (it depends on the preferences of the team). Prototypes are mock-ups of the screens and windows of an application that represent the front end of the system. Note that in this method there is no role distribution. So, any member of the team can ask or draw a prototype without distinction. The prototypes are validated with the client through interviews.

The process during the interviews ([41, 42]) is as next:

- In the first meeting, the members of the team take notes and draw some quick sketches from the comments of the client on a paper.
- After this first meeting, the team elaborates a prototype (in a paper or with a graphical tool similar to Balsamiq [43]).
- In a second meeting, the members of the team validate this prototype with the client. This prototype is also used to extract new requirements or to specify other requirements that were not so clear in the first meeting. During the interview, the team can make corrections on the prototype.
- After the second meeting, the members of the team rebuild the prototype in detail with all the corrections and extensions extracted from the second meeting (in a paper or with a graphical tool).
- If needed, in the third and fourth meetings the members of the team validate the new prototypes again. In these last meetings the client checks the prototypes and proposes the last corrections to inconsistencies.

- At the end, with the last corrections over the prototypes, the team elaborates the requirements specification document.

4 Experiment Design

4.1 Goal

The goal of this experiment is to compare different requirements elicitation methods based on interviews, with the purpose of identifying the pros and cons of each one. The analysis is based on the elaboration of the requirements specification document. The experiment is conducted from the perspective of researchers and practitioners interested in investigating how much better each method is regarding the other ones.

4.2 Experimental Subjects

The *participants* are undergraduate students of two degrees of the Universitat de València (UV, Spain): Computer Science Engineering (CE) and Telematics Engineering (TE), in which two of the authors are instructors. The experiment is conducted in Spanish, the native language of all the subjects. Subjects are enrolled students in the course of Software Engineering, which is taught in both degrees in the same way and by the same instructors. The participation in the experiment was mandatory since it was an evaluable task as part of the course, among others. The pedagogical value for students is that they learn about how to elicit requirements through real interviews. They are already accustomed to eliciting requirements from a textual description, but this experiment is the first experience in eliciting requirements through interviews with persons. All the subjects gave explicit consent to analyze the data at the beginning of the experiment [44], so we could analyze data gathered from of all of them. Software Engineering is taught in the fourth semester of CE degree and the fifth semester of TE degree. The topic of this subject is mainly UML: analysis, design, and implementation from models. Within this subject, participants have to develop a project from scratch, from requirements elicitation to the implementation of the final code. The part to elicit requirements is used as the target of study in our experiment. One instructor plays the role of client, and the students (subjects) play the role of developers, who have to elicit the requirements and create a textual requirements document in IEEE 830-1998 standard [9]. All the process of elicitation is conducted through interviews between the client and subjects. Table 1 shows the number of subjects recruited in each replication. Note that, in general, teams were composed of 6 members but there were a few teams with less members due to unpaired students in the class. The family of experiments is composed of two replications in CE and two replications in TE. Subjects work in teams of 6 members at maximum. In our experiment, each team is an experimental unit, since we extract data to analyze from each team.

Table 1. Subjects per replication

Degree	2015-2016		2016-2017		2017-2018	
CE	Team 1	5	Team 1	6		
	Team 2	6	Team 2	6		
	Team 3	5	Team 3	6		
	Team 4	6	Team 4	5		
	Team 5	6	Team 5	6		
	Team 6	5	Team 6	6		
	Team 7	5	Team 7	4		
	Team 8	3	Team 8	4		
	Team 9	4	Team 9	2		
	TOTAL	**45**	**TOTAL**	**45**		
TE			Team 1	6	Team 1	5
			Team 2	6	Team 2	6
			Team 3	6	Team 3	6
			Team 4	6	Team 4	6
			Team 6	6	Team 5	6
			Team 7	6	Team 6	4
			TOTAL	**36**	**TOTAL**	**33**

Subjects are around 20 years old in CE and 21 years old in TE that did not attend any previous course in which requirements elicitation process is explained. However, subjects have previous knowledge in programming both in

Java and C++. Before starting the interviews, the instructor provides self-elaborated material about the different methods to elicit requirements; this way, subjects know how to apply them in a practical context as the experiment. In order to ensure that all the subjects have enough knowledge of the requirements elicitation method, the subjects had to fill in a small questionnaire about the method to apply. This way, we can ensure that the subjects had read the material of the method. Only a few teams (Team 5 and Team 8 in TE course 2016-2017) were not considered to participate in the experiment due to the low mark in the exam of the method.

4.3 Research Questions and Hypothesis Formulation

We aim to compare the requirements elicitation methods quantitatively. In the existing literature, there are works that focus on dealing with requirements through textual requirements elicitation, such as the work of Dieste et al. [10], and there are also works that propose evaluating Use Case diagrams [45]. Both types of evaluations allow quantitative analysis. In our experiment, we have chosen the use of a textual specification, since we are avoiding syntactical problems that appear when analysts work with Use Cases. A textual description allows us to focus the evaluation on the semantics of the requirements.

In the work of Dieste et al. [10], there is a list of variables to evaluate elicitation requirement methods; we have extracted our research questions from that list:

- **RQ1:** *Is quantity of requirements affected by the requirements elicitation method?* By quantity we mean how many requirements are elicited, independently of their quality. The null hypothesis tested to address this research question is: *H_{01}: The quantity of requirements is similar among JAD, Paper Prototyping and Unstructured Interviews.*
- **RQ2:** *Is time to elicit requirements affected by the requirements elicitation method?* By time we mean the number of minutes to prepare the interview, the minutes spent within the interview and the minutes spent in the elaboration of the requirements specification document. The null hypothesis tested to address this research question is: *H_{02}: The time to elicit requirements is similar among JAD, Paper Prototyping and Unstructured Interviews.*
- **RQ3:** *Is diversity of requirements affected by the requirements elicitation method?* By diversity we mean the number of requirements that are overlapped. The null hypothesis tested to address this research question is: *H_{03}: The diversity of requirements is similar among JAD, Paper Prototyping and Unstructured Interviews.*
- **RQ4:** *Is completeness of requirements affected by the requirements elicitation method?* By completeness we mean the percentage of requirements covered with the requirements specification. The null hypotheses being tested to address this research question is: *H_{04}: The completeness of requirements is similar among JAD, Paper Prototyping and Unstructured Interviews.*
- **RQ5:** *Is quality of requirements affected by the requirements elicitation method?* According to IEEE vocabulary [46], quality *is the ability of a product, service, system, component, or process to meet the client or user needs, expectations, or requirements.* In our experiment, we have considered by quality the number of requirements properly described. The null hypotheses being tested to address this research question is: *H_{05}: The quality of requirements is similar among JAD, Paper Prototyping and Unstructured Interviews.*
- **RQ6:** *Is performance of the elicitation process affected by the requirements elicitation method?* According to IEEE vocabulary [46], performance is *the degree to which a system accomplishes its designated functions within given constraints*. In our experiment, we deal with performance as the effort that the analyst invested in identifying the list of requirements. The null hypotheses being tested to address this research question is: *H_{06}: The performance to elicit requirements is similar among JAD, Paper Prototyping and Unstructured Interviews.*

4.4 Factors and Treatments

This section defines the factors (independent variables used to see the effect on the response variables [47]) and the treatments (alternatives of a factor [47]). Our experiment is based on one factor: the requirements elicitation method. The control treatment is the method Unstructured Interviews, while JAD and Paper Prototyping are the other treatments we aim to compare.

As Section 3 describes, Unstructured Interviews is a method where each member of the team can play different roles (interviewer, writer, etc.). There is not a procedure to guide the interview and the artifact to report the requirements depends on the analysts' decision. Regarding JAD, this method is different from Unstructured Interviews in the fact that each member of the team has a specific role to play during the interview. Each role has tasks to guide the interview or to report the results. Regarding Paper Prototyping, there are no specific roles as in the Unstructured Interview. However, there is a specific artifact (as described in Section 3) used to report the requirements and to get feedback from the client: the prototype.

4.5 Response Variables and Metrics

This section describes the response variables (dependent variables that show the effect studied in the experiment caused by the manipulation of factors [47]). Several of these metrics are based on a list of requirements elaborated by the two instructors. This list is named "instructors' solution" and was developed in consensus, putting together from scratch all the requirements that the client should require to the team of developers during the interviews. Other metrics are based on events that happened during the interviews. In order to avoid interference between the requirements elicitation process and the measure of the variables, interviews were recorded by each team and metrics related to the interviews were calculated afterward.

RQ1 requires a variable to measure **the number of requirements**. There are two types of requirements, functional and non-functional. According to Sommerville [5], functional requirements define the behavior of the system (how the system should react to inputs and the services it should provide). Chung [48] defines non-functional requirements as a software requirement that does not describe what the software will do, but the properties and constraints under which the software will operate. In our target study, we have used three metrics for RQ1: the number of functional requirements, the number of non-functional requirements and the addition of both functional and non-functional requirements.

RQ2 requires a variable to measure the **time** spent to elicit requirements. There are several metrics to measure the time in our experiment: time to prepare the interview, time to conduct the interview, time to report the requirements in a requirements specification document, and the addition of all the previous times. Each method can have different values for each metric of time, we aim to study which one is the fastest and which one is the slowest. Note that we measure the time for the whole requirements elicitation process. Teams used from 2 to 4 interviews depending on their necessity; these times were summed up and considered to be the time spent to conduct the interview.

RQ3 requires a variable to measure the **diversity** of requirements elicited. By diversity we mean that each requirement deals with specific characteristics and there is no overlapping among requirements. The metric for diversity is the number of overlapped requirements. By overlapped requirements we mean the requirements which are partially covered by other requirements in the same specification document.

RQ4 requires a variable to measure the **completeness** of the list of requirements identified. By completeness we mean the percentage of identified requirements by each experimental unit that agree with the requirements of the instructors' solution. The metric for completeness is the percentage of identified requirements of the total amount of requirements.

RQ5 requires a variable to measure the **quality** of the list of requirements. Quality means how much accurate is the list of requirements. The metric we used to measure quality is the number of requirements that agree with the instructors' solution plus the number of requirements that are not in the instructors' solution but that are elicited properly, minus the number of requirements that are in the instructors' solution but that are not elicited properly, all divided by the total amount of requirements in the instructors' solution. In summary, this variable represents the percentage of requirements that are not specified wrongly by the experimental units, independently whether or not they are specified in instructors' solution. By a wrong requirements specification, we mean that the requirement was identified but its description was not precise or contradicted some requirements of the instructors' solution.

Table 2. Summary of RQs, hypotheses and response variables

RQs	Hypotheses	Response Variables	Metric
RQ1	H_{01}	Number of requirements	Number of functional requirements Number of non-functional requirements Number of functional requirements + non-functional requirements
RQ2	H_{02}	Time	Time to prepare the interview Time in the interview Time to prepare the requirements document Overall time
RQ3	H_{03}	Diversity	Number of overlapped requirements
RQ4	H_{04}	Completeness	Percentage of requirements that agree with the requirements of the instructor's solution
RQ5	H_{05}	Quality	(requirements that agree with instructors' solution + requirements that are not in instructors' solution but that are properly specified – requirements wrongly specified) / total amount of requirements in instructors' solution
RQ6	H_{06}	Performance	Number of questions Number of relevant questions Efficiency (quality time ratio)

RQ6 requires a variable to measure the performance of the list of requirements identified. Performance is the effort needed to identify the requirements. We propose measuring this effort through three metrics: Number of questions during the interview; Number of relevant questions (a subset of all the questions); Efficiency, measured as the ratio of quality per time.

Table 2 shows a summary of the research questions, with the hypotheses to answer them, the response variables to extract data, and the metrics used to conclude results. Note that the metrics are calculated by experimental units by their own. Once the experimental units have completed the list of requirements, the instructors provide the instructors' solution and subjects apply the metrics through a spreadsheet.

4.6 Differences and Similarities among Replications

This section describes the characteristics of the experiments that remain stable among the different replications and the others that change. What remains unchanged for all the replications are: factor and treatments, metrics, response variables, experimental design, instruments, experiment procedure, and instructors. Next, we describe the changes among replications and the reasons for such changes:

- Experimental object: we used a different experimental object each academic year. Note that the same problem was used in both degrees (CE and TE) during the same academic year. This change is justified due to the need to extract conclusions independently of the experimental objects used in the experiments. This way, we can ensure that results do not depend on a specific object. All the experimental objects used in the three years of replication of the experiment were similar in difficulty (number of function points) and application domain (management information systems with CRUD – Create, Read, Update, Delete – operations).
- Sample size: As Table 1 shows, there is a slight variation in the sample size for each replication. Instructors could not control this characteristic, it only depended on the number of subjects per course, which varies by year. Note that this does not affect the experiment.
- Subjects profile: we have 2 replications in CE degree and other 2 replications in TE degree. This was done to ensure that results do not depend on a specific profile of the subjects. Note that although degrees are different and CE is scheduled in the fourth semester while TE is scheduled in the fifth semester, the content of the course is the same in both of them. Differences of both degrees depend on other courses different from the course used in the experiment. Therefore, differences between the subjects' profiles of TE and CE are minimum.

As a summary, we conclude that all the replications of the family share most of the experimental characteristics. Changes only arise to avoid possible threats related to the generalization of results and due to the random effect of the number of students' registrations each course.

4.7 Experiment Design

This section describes the design we followed in our experiment. We have a between-subject effect, since we examine differences between subjects. Each experimental unit only receives one treatment. As Table 3 shows, the design is one factor, three treatments [47]. We have three different groups of experimental units (G1, G2, G3); each group only received one treatment. The assignment of a team (experimental unit) in a group was done randomly and balanced. Table 4 shows the assignment of treatment per team. The experimental problem to solve is the same per replication, independently of the treatment.

Table 3. One factor, three treatments

	Problem
Unstructured Interview	G1
JAD	G2
Paper Prototyping	G3

The main drawback of using this design is that we are not using the maximum sample size, since we are dividing the sample into three groups. A repeated measure design would have avoided this issue, but we could not apply repeated measures since the time to conduct the experiment was limited, and the application of the three treatments to every team would have increased the required time more than the available one. Another drawback is that, from a teaching point of view, each team only learns in practice one requirements elicitation method. Even though only one requirements elicitation method is assigned per team, all students have the chance to review the material of the other methods to learn them from the didactic point of view. Moreover, at the end of the experiment, all teams present a summary of their elicited requirements to verify the metrics used in the experiment. This summary includes a short description of the elicitation method used per each team. This way, all subjects can learn all methods theoretically, even

though they can only put into practice one. A threat of this design is that we are using only one experimental problem per replication, reducing the generality of the results. We could not use more problems in order to simplify the role of the client played by one instructor. In order to minimize this threat of the design, we changed the problem in each replication, working with several problems throughout the whole family of experiments. This change aims to mitigate the threat of validity regarding the generalization of results, improving their validity. All problems maintain a similar size but their domain is different.

The main advantage of this design is that the conditions for the three treatments are exactly the same, which optimizes the comparison among treatments. Another advantage is that since only one treatment is applied per team, we are avoiding the learning effect.

Table 4. Assignment of treatment per team

Degree	Treatment	2015-2016	2016-2017	2017-2018
CE	Unstructured Interview	Team 3	Team 8	
		Team 4	Team 4	
		Team 8	Team 9	
	JAD	Team 5	Team 1	
		Team 6	Team3	
		Team 7	Team 6	
	Paper Prototyping	Team 1	Team 2	
		Team 2	Team 5	
		Team 9	Team 7	
TE	Unstructured Interview		Team 6	Team 1
				Team 4
	JAD		Team 1	Team 2
			Team 3	Team 5
			Team 7	
	Paper Prototyping		Team 2	Team 3
			Team 4	Team 6

4.8 Experimental Objects

We have used a different experimental object (problem) in each replication. Experimental units have to schedule interviews with the client (role played by one instructor) and elicit requirements from these meetings. At the end of the process, each experimental unit has to prepare a requirements specification document with all the identified requirements according to IEEE 830-1998 standard [9]. This document is the artifact used to compute our metrics. Next, we describe each problem.

- Replication 2015-2016 in CE: This problem consists of a system to manage a hotel. There are two actors in the system, the receptionist and the administrator. The receptionist can query the list of rooms for a specific date, make a reservation, cancel a reservation, register the check-in and register the check-out. Regarding the administrator, this actor can do the same activities as the receptionist; in addition, there are specific actions she/he can do, such as, modify the price of the rooms and modify the discounts depending on the guests' age. Appendix A reports the list of the requirements that the instructor expressed during the interviews, interpreting the role of the client. Note that these requirements are part of the instructors' solution to be homogeneous in all the meetings, but they are not disclosed to students. The size of this problem is 481 function points (small problem).
- Replication 2016-2017 in CE and TE: The same problem was used in both degrees during this course. The problem consists of a system to manage the sale of flights. There are three actors in the system: the client, the employee and the supervisor. The client can only buy flights according to dates and departures/destinations. The employee can change the state of a flight to boarding, cancel a flight, delay a flight, and create a flight. The supervisor can perform the same activities as the employee as well as others more specific: create an itinerary between two cities, remove an itinerary and modify the flight prices. Appendix B has the list of requirements that were part of the instructors' solution. The size of this problem is 637 function points (small problem).
- Replication 2017-2018 in TE: This problem consists of a system to manage a cruiser with tourists. There are four actors in the system: the director, the concierge, the waiter, and the employee. The director can create a new itinerary between two cities, create a cruiser, query the cruisers in which a ship participates, query how many people entered in a specific restaurant of the cruiser on a specific date. The concierge only can register tourists in an excursion. The waiter registers when a tourist enters in a restaurant of the cruiser. Finally, the employee can buy tickets of the cruiser and cancel reservations. Appendix C reports the list of requirements that were part of the instructors' solution. The size of this problem is 697 function points (small problem).

All the problems have similar difficulty and focus on CRUD operations. Usually, some conditions apply to the operation, for example, to cancel a flight you must charge an economic punishment if the departure date is less than 2 days. Experimental units had to elicit requirements of the problem and write down them in a requirements specification document. Metrics of the experiment (described in section 4.5) are applied to this document. The similarity in the difficulty of the problems makes it possible to compare the metrics of all of them.

4.9 Experiment Procedure

This section describes how the experiment was conducted. Fig 1 shows a graphical summary from the team formation until the application of the metrics to report the data. Next, we describe each step.

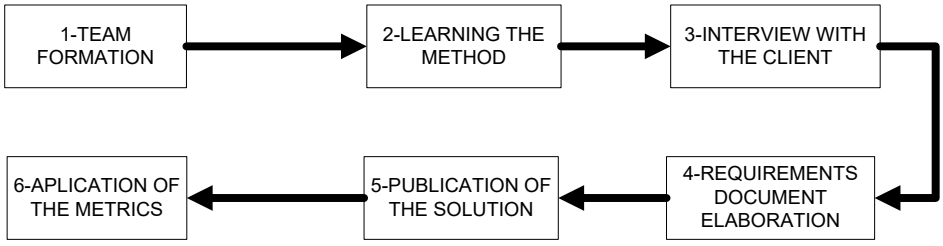

Fig 1. Summary of how the experiment was conducted

- **Step 1, Team Formation**: subjects are grouped in teams of 6 persons at most (the exact number of members for each team is reported in Table 1). The instructor assigns the members to each team randomly. Each elicitation requirements method is applied to each team randomly but ensuring that all the methods are balanced.
- **Step 2, Learning the Method**: The instructor provides the material necessary to learn the assigned requirements elicitation method. The two instructors elaborated the material for the three methods, ensuring that the level of detail of each method was similar to that of the others. Each subject learns the method on her/his own with the support of the instructor. Only subjects that pass an exam with questions regarding the requirements elicitation method were recruited for the experiment.
- **Step 3, Interview with the Client**: the instructor plays the role of the client while the subjects play the role of developers. Requirements elicitation must be done through interviews between both roles. Each team must arrange the meetings with the client in advance; interviews are arranged by the team (interviews are not shared among several teams). The client participates only in interviews when a team asks for it. In general, each meeting lasts around 30 minutes, and most teams used 3 interviews (the maximum number of possible meetings was 4). During each interview, the client has to explain all the requirements, answering precisely each question asked by the developers. Each team applies the particularities of the requirements elicitation method that it must use. The instructor playing the role of the client during the interviews also checked that each team had correctly understood the method they studied and how to apply it.
- **Step 4, Requirements Document Elaboration**: With all the information collected with the interviews, each team must elaborate a document that gathers all the requirements (functional and non-functional). The document is a list of requirements to be considered when the system is developed. At the end of this step, the document with the format of IEEE 830-1998 standard [9] is submitted to the instructors with no possibility to change it.
- **Step 5, Publication of the Solution:** The instructors have elaborated a document that includes all the requirements. The existence of this document is previous to the interviews; this way the instructor that plays the role of client ensures that the answers to the questions are homogeneous. This solution is shared among all the teams such a way they can compare its solution, submitted in Step 4, versus the instructors' solution.
- **Step 6, Application of the Metrics:** The instructors have elaborated a spreadsheet to compare the list of requirements elicited by the subjects versus the requirements of the instructors' solution. This spreadsheet includes fields with formulas that calculate experimental metrics. Subjects only have to fill in the data required to apply the metrics. We collect a spreadsheet per team.
- **Step 7, Verification of computed metrics:** The metrics computed by teams applying the same methods are compared in a plenary session which lasted 2 hours. The metrics are checked showing the results of each team to all the subjects and instructors. When any subject or instructor considers that some results were not usual, they are investigated deeply by the instructors in collaboration with the team that applied the metric. In case of mistakes caused by a misunderstanding of the metrics, they are re-computed by the instructors, who ensure that the metrics are applied properly. On average, we found around 1 mistake per team.

4.10 Threats to Validity

This section discusses threats to validity of the experiment in order to know to what extent the results are not biased. According to [49], we classify the threats into 4 groups:

- **Conclusion validity**: This type of threat deals with the ability to draw the correct conclusion about relations between the treatment and the outcome. The experiment may suffer the following threats of this type. *Low statistical power,* which means that the number of subjects is not enough to reveal a true pattern in the data. We recruited 30 experimental units (167 subjects) adding the different replications of our family of experiments. Even though there are works such as [50] that state that a sample size of 30 is enough to be considered significant, with a simulation with our design in G*Power [51] we conclude that we need a sample of 102 experimental units to get a large power. Note that we do not have a repeated measures design, so we are losing power applying one treatment per team (we split the sample size per treatment). We had to gather subjects in teams since how groups work with elicitation methods is also a target of our study. This reduces experimental units considerably. Another threat that our family of experiments may suffer is *Fishing,* which means that the instructors are looking for a specific result. This threat has been minimized since there is not a specific treatment proposed by the instructor, leading to think that the family of experiments has not been designed to benefit any treatment. Another threat that appears in the design is *Reliability of measures,* which means that the measures used in the statistical analysis may include mistakes. In our design, where each team applies the metrics to get the measures, this threat is probable. In order to minimize this threat, we reviewed all the measures in a plenary session with all the subjects asking them to review the data when we identified anomalous data. Another threat that our design may suffers is *Reliability* of *treatment implementation*, which means that the treatment could be not implemented properly. In our design, subjects learn the method to elicit requirements and apply it through interviews. It is possible that some mistakes in the application of the method arise. In order to minimize this threat, we took two actions. First, we examined all the subjects with an exam about the method they had to apply before conducting the experiment. Only teams that passed the exam were recruited for the experiment. We had two teams that we rejected in the whole family of experiments. Second, the instructor that played the role of client in the interviews ensured that the method was properly applied. Another threat that our design suffers is *Random irrelevancies in experimental settings*, which means that elements outside the experimental setting may affect the results. Since each experiment of the family was conducted in different months, we cannot ensure that the contextual circumstances were the same for all the subjects.
- **Internal validity**: This type of threat deals with influences that can affect the factor concerning causality. The design may suffer the threat *History,* which means that subjects could pass information to other subjects in next replication. We solved this threat changing the problem in each replication. Another threat that our design may suffer is *Instrumentation*, which means that the artifacts used in the experiment could affect the results. The metrics are applied through a spreadsheet that helps to gather all the data. We have minimized this threat through a plenary session in the last class with all the subjects to check the data written in each spreadsheet. Another threat is *Interactions with selection*, which means that different groups of subjects may have a different behavior with the same treatment (different maturity, background, etc.). We have tried to minimize this threat describing the treatment in the same way to all the subjects of the family of experiments. Another threat is *Diffusion or imitation of treatments,* which appears when a group of subjects applies some features of other treatment different to the one assigned. This could happen, for example, when groups of Unstructured Interviews used some features of JAD or Paper Prototyping. We tried to minimize this threat checking how each team applied the method in the interviews. In case the instructor that played the role of client identified that any feature was not part of the assigned method, this mistake was solved. Another threat is *Resentful demoralization*, which means that some treatments can be more motivating than others. In our case, teams with Unstructured Interviews could be less motivated than teams with JAD and Paper Prototyping.
- **Construct validity**: This type of threat concerns generalizing the result of the experiment to the concept or theory behind the experiment. Our design may suffer the threat *Mono-method bias*, which means that using a single type of measures involves a risk. We based our analysis on metrics that are not double-checked. We tried to mitigate this threat through a plenary session which took place in the last class, during which we looked for mistakes in the metrics application. Another threat that may suffer our design is *Interaction of testing and treatment*, which means that testing itself is part of the treatment. In our family of experiments, experimental units had to test the results of applying the treatment. This involves that experimental units were more aware of the errors they made, and thus being able to reduce them. Another threat that may appear is *Evaluation apprehension,* which means that some people are afraid of being evaluated. This behavior could lead to writing down better values in the metrics than the real ones. We have tried to mitigate this effect through the plenary sessions in which we compared the metrics among all the teams, correcting those that were wrongly computed.

- **External validity**: This type of threats concerns with the extent to which it is possible to generalize the findings of the study. Our family of experiments suffers the threat *Interaction of selection and treatment,* which means the effect of having a subject population not representative of the population we want to generalize. In our family of experiments, we cannot ensure that results can be generalized to people with different profiles of our sample. However, we are mitigating this threat through the recruitment of subjects from two different degree programs, with two different profiles, during several courses. This ensures that there are subjects with a profile more specialized in software development (CE) and subjects with a profile more specialized in networks management and hardware (TE). Another threat that may suffer the experiment is Confounding factors, which happens when results may be affected by other factors beyond the experiment. We are comparing elicitation methods that are measured through a requirements elicitation document. How each method is operationalized into the document may be affected by other factors such as skills or subjects' background, among others. Another threat that appears is *Interaction of settings and treatment,* which means the effect of not having the experimental setting or material representative of industrial practice. Note that even though we have done our best to implement a context as real as possible through interviews, at the end we are conducting an experiment in an academic environment.

4.11 Data Analysis

Data have been analyzed through descriptive statistics and statistical tests, all of them through the software SPSS 20.0. Raw data can be seen in Appendix D. For descriptive statistics we have used box-and-whisker plots to analyze possible differences among treatments regarding medians, first quartile, and third quartile. For the statistical tests, we have used the Kruskal-Wallis test for independent samples [52]. If the p-value is less or equal[1] to 0.05, we assume that there are significant differences between treatments. The choice of this test is based on the fact that data do not follow a normal distribution and we have 30 experimental units in total (9 with Unstructured Interviews, 11 with JAD and 10 with Paper Prototyping) which is not a big number to work with a parametric test with a between-subjects design, since samples units are divided by three in the analysis. The Kruskal-Wallis test has two assumptions: independent experimental units and homoscedasticity (homogeneity of variance). The first assumption is satisfied since there is no relationship among the observations in each treatment. The second assumption has been verified by applying the Levene's test. All p-values obtained with the Levene's test are bigger than 0.05, which means that there are no differences among population variances.

For those variables with significant differences among treatments, we calculate their effect size. This calculus yields the magnitude of such difference. For the effect size, we have used eta square, which is interpreted as follows: values around 0.01 means a small effect; values around 0.06 means medium effect; values around 0.14 means a large effect. We have used G*Power [51] to evaluate the power of the statistical test. For a large effect size in a between-subjects design, we need a sample size of 102 experimental units. In our experiment, we have 30 experimental units, which implies a low power. This means that some of the null hypotheses might not be rejected due to the small sample size.

Note that all the metrics have been applied by the experimental units by their own, which is a threat for the validity of the data. Although we have tried to minimize this thread with the supervision of the instructor during the data extraction and through a final plenary session to check the data with the own subjects, we cannot ensure that all the metrics were applied properly. In order to avoid that a wrong application of the metrics could affect our results, we have removed outliers from our data repository [53]. For this removal, we have drawn the box-and-whisker plots for each response variable and we have removed the points that appeared as outliers in such graphical plots.

5 Results

This section deals with the analysis extracted through the metrics. Next, we structure the results by response variable; the names of the variables used in SPSS appears in brackets. The variable **Number of requirements** is calculated by means of three metrics: number of functional requirements (number_functional_requirements), number of non-functional requirements (number_non_functional_requirements), and the addition of functional plus non-functional requirements (number_of_requirements).

Fig 2.a shows the box-and-whisker plot only for functional requirements. Medians show that the number of functional requirements got through Paper Prototyping is higher than with the other two methods. First and third quartiles are better for Unstructured Interviews and Paper Prototyping. Box-and-whisker plot for number of non-functional requirements can be seen in Fig 2.b: in this case, as values of median, first, and third quartiles show, the method that allows identifying more non-functional requirements is JAD, while Paper Prototyping is the worst. Fig 3 shows the box-

[1] Traditionally, only p-values lower to 0.05 are considered as significant. We are also considering significant values equal to 0.05. This choice depends on how conservative we want to be with the analysis. We think that with this choice we are not rejecting significant results very close to 0.05 but lower, since SPSS rounds results.

and-whisker plot for the addition of both functional and non-functional requirements. There is a slight difference among the means: Paper Prototyping identifies the highest number of requirements, next in number of requirements identified is JAD and finally Unstructured Interviews. First and third quartiles yield better values for Unstructured Interviews.

Table 5 shows the results of the statistical analysis. From all the analyzed metrics, only number_of_requirements got a p-value less than 0.05, which means that the total amount of requirements have significant differences among treatments. Looking at the box-and-whisker plot in Fig 3, we see that Paper Prototyping yields the highest number of requirements. These differences are moderate according to the effect size of 0.08. As a conclusion of our analysis, since p-value < 0.05 we state that we can reject *H_{01}: The quantity of requirements is similar among JAD, Paper Prototyping and Unstructured Interviews* when we consider functional and non-functional requirements altogether. This result means that **Paper Prototyping is the most suitable method to identify as many requirements as possible, independently of the correctness** of such requirements. This might be due to the fact that graphical prototypes help more in the identification of requirements than the combinations of having different roles in the team as in JAD or running Unstructured Interviews.

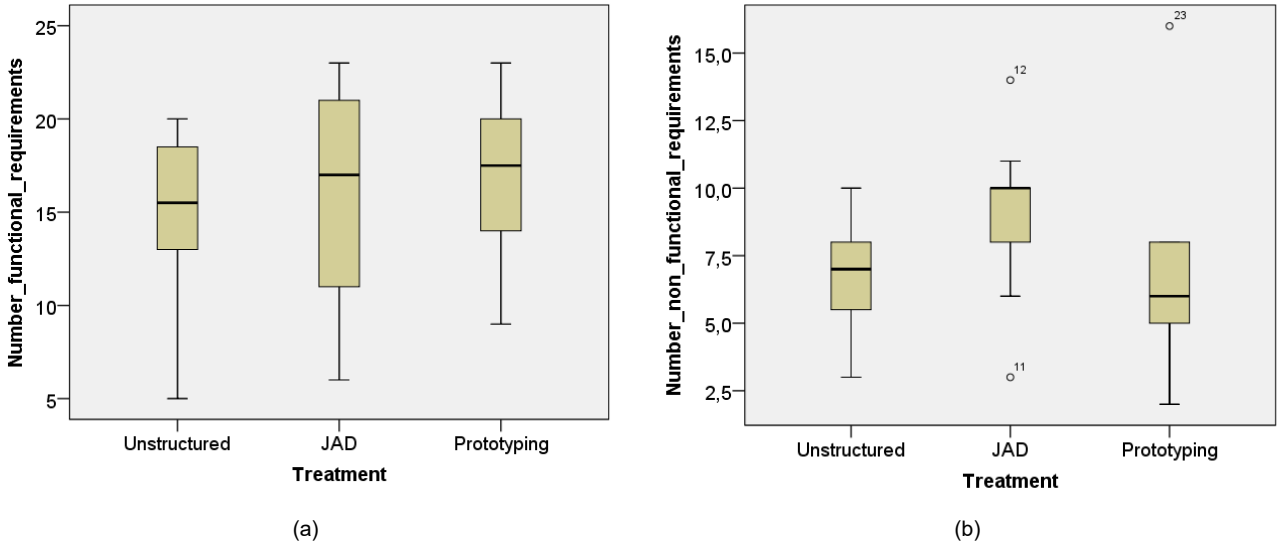

Fig 2. box-and-whisker plot for (a) number of functional requirements and (b) number of non-functional requirements

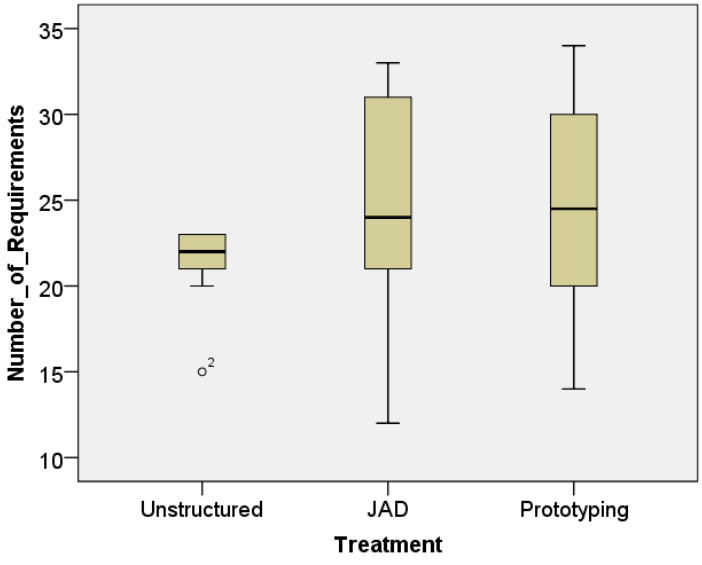

Fig 3. Box-and-whisker plot for number of requirements

Table 5. Statistical analysis for metrics of number of requirements

	Mean	Std. Deviation	p-value	Effect size
Number_functional_requirements	16.04	5.01	0.68	-
Number_non_functional_requirements	7.54	3.25	0.10	-
Number_of_requirements	23.48	6.30	0.02	0.08

Response variable **Time** is calculated through several metrics: time spent in preparation of the meetings (Time_to_prepare_interviews), time spent within the interview (Time_spent_in_interviews), time to write down the requirements document (Time_to_report), the addition of all the other times (OverallTime). Fig 4.a shows the box-and-whisker plot for Time_to_prepare_interviews. JAD is the method with the highest median, which means that it is the method that requires more time to prepare an interview. The method that needs less time is Unstructured Interviews, which have the lowest median, first and third quartiles. Regarding Time_spent_in_interviews, as shown in Fig 4.b, there are not many differences among the medians of the three methods. The first and third quartile with less time are for Unstructured Interviews. Fig 5.a shows the box-and-whisker plot for Time_to_report. JAD is the method with the highest time, since it has the highest median, first and third quartiles. Unstructured Interviews is the method with the lowest time. Fig 5.b shows the box-and-whisker plot for OverallTime. Again, JAD is the method with the highest time and Unstructured Interviews is the method with the lowest one. Note that the method with less variability of time is Paper Prototyping, since median, first and third quartiles are close to each other.

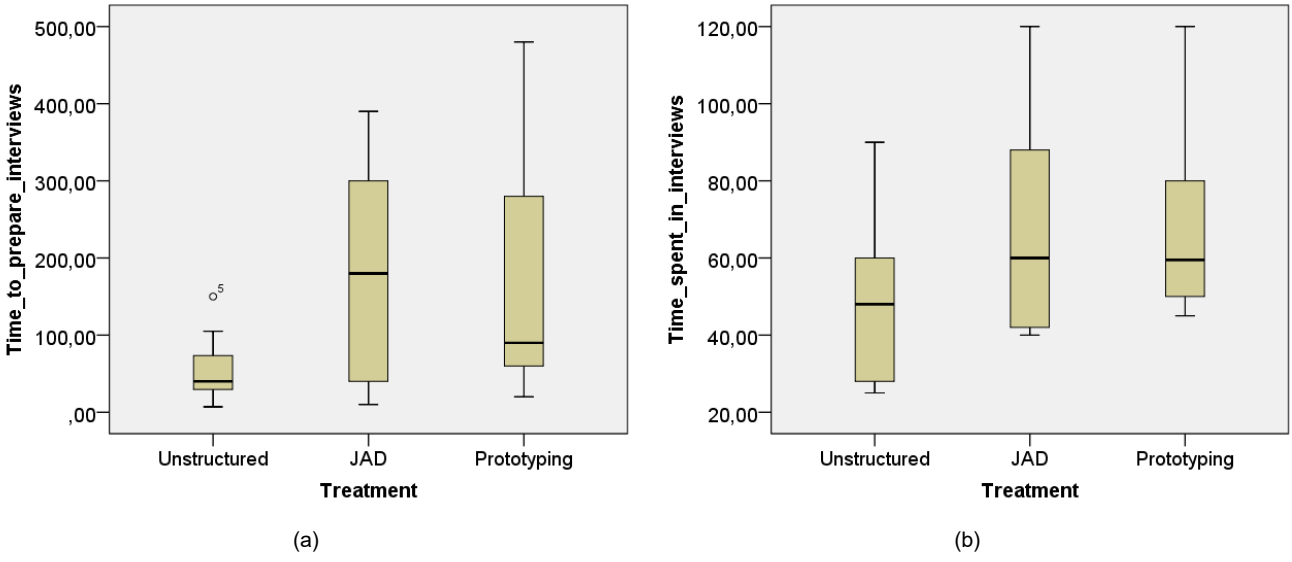

Fig 4. Box-and-whisker plot for (a) time spent to prepare the interview and (b) time spent within interviews

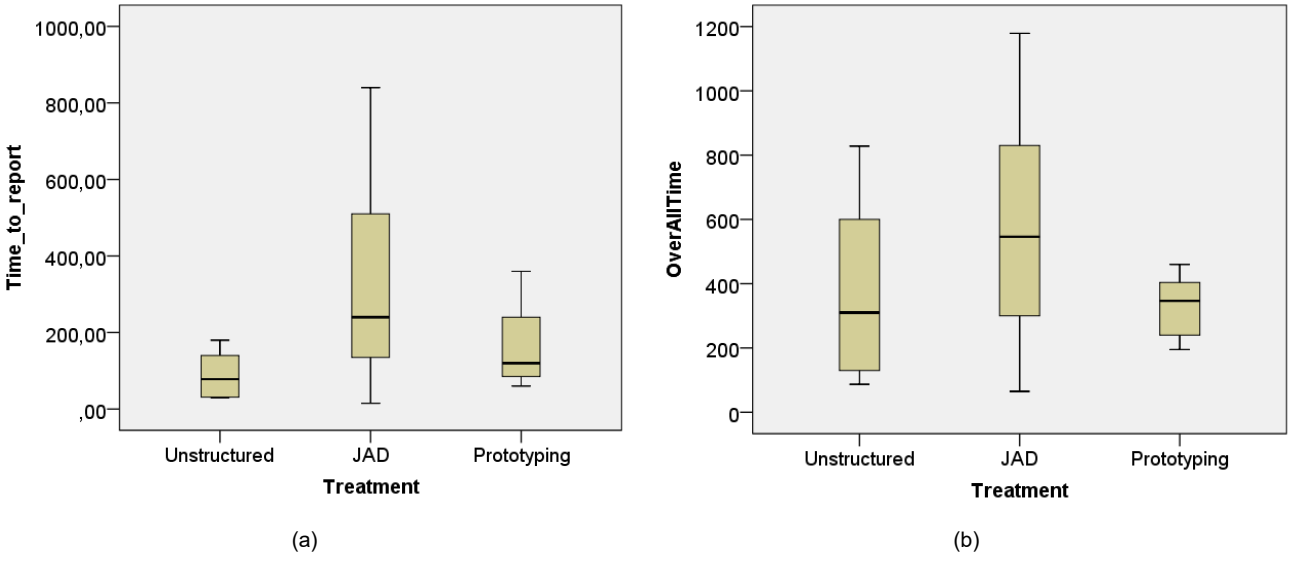

Fig 5. Box-and-whisker plot for (a) time to write down the requirements and (b) addition of all the times

Table 6 shows the results of the statistical analysis for the metrics for Time. The only metric with significant differences is Time_to_report, with a large effect. According to Fig 5.a, the method that requires less time to write down the requirements was Unstructured Interviews, and the method that requires more time was JAD. As a conclusion, since p-value<0.05 we can reject *H_{02}: The time to elicit requirements is similar among JAD, Paper Prototyping and Unstructured Interviews* when we measure the time to write down the requirements in a document. **Unstructured Interviews is the fastest method and JAD is the slowest**. Analyzing the other metrics of time, we see that this pattern of differences is repeated in all of them, although these differences are not significant. In general, JAD is the method that requires more time for all the steps of the requirements elicitation process. Note that the time to draw the prototypes in Prototyping is included in the metric time_to_prepare_interviews, which yields less time than the time required in JAD. This means that the time spent in JAD for the coordination among the different roles in the team is longer than the time needed for drawing prototypes. These results show a relationship between number of requirements and time to report. Unstructured Interviews identify fewer requirements and the time to report them is obviously less. Similarly, Unstructured Interviews requires less time to prepare interviews.

Table 6. Statistical analysis for metrics of time

	Mean	Std. Deviation	p-value	Effect size
Time_to_prepare_interviews	145	137.62	0.11	-
Time_spent_in_interviews	60.37	24.47	0.31	-
Time_to_report	209.03	210.79	0.05	0.22
OverallTime	435.67	288.84	0.31	-

Response variable **Diversity** has only one metric based on the number of overlapped requirements. Fig 6 shows the box-and-whisker plot for this metric. The method with less overlapping is JAD, since it has the lowest median and first quartile. Unstructured Interviews is the method with the highest overlapping, as it has the highest median, first and third quartiles.

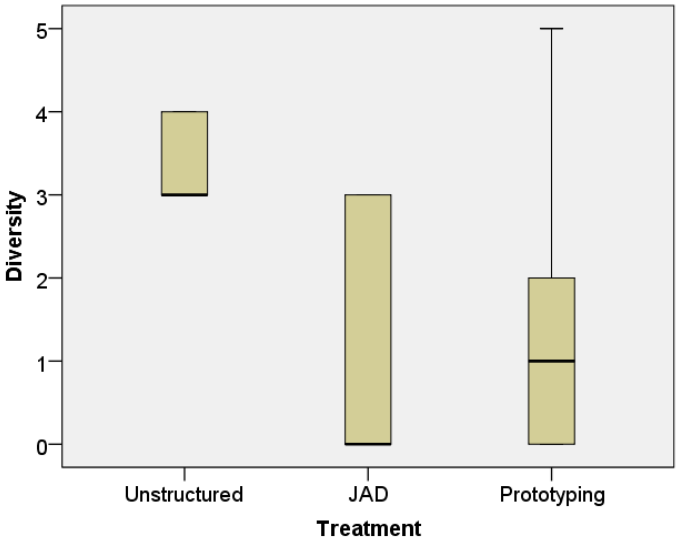

Fig 6. Box-and-whisker plot for number of overlapped requirements

Table 7 shows the statistical results of the metric for Diversity. P-value shows that there are significant differences among the methods. Considering Fig 6, we can state that significantly, JAD is the method producing fewer overlapping requirements and Unstructured Interviews is the method yielding more.

As a conclusion, since p-value< 0.05 we can reject *H_{03}: The diversity of requirements is similar among JAD, Paper Prototyping and Unstructured Interviews*. This means that when there are different roles in the team and each role has specific responsibilities, the number of overlapped requirements is low. This might be because the roles separation helps to focus the responsibilities of each member within the team, so overlapping among tasks to elicit requirements is low. The absence of roles and the lack of specific artifacts to help in the elicitation process could be the reason for the

high level of overlapping that Unstructured Interviews yields. Without roles separation, more than one member could report the same requirement, which involves the duplication of the same requirement.

Table 7. Statistical analysis for the metric for Diversity

	Mean	Std. Deviation	p-value	Effect size
Number of overlapped requirements	1.75	1.7	0.03	0.29

Response variable **Completeness** is calculated as the percentage of requirements included in the instructor's solution that were identified. Fig 7 shows the box-and-whisker plot for this metric. There is no difference between the medians for the three methods, while the first and third quartiles are better for Paper Prototyping. This means that even though there are not important differences among treatments, Paper Prototyping yields a little better result.

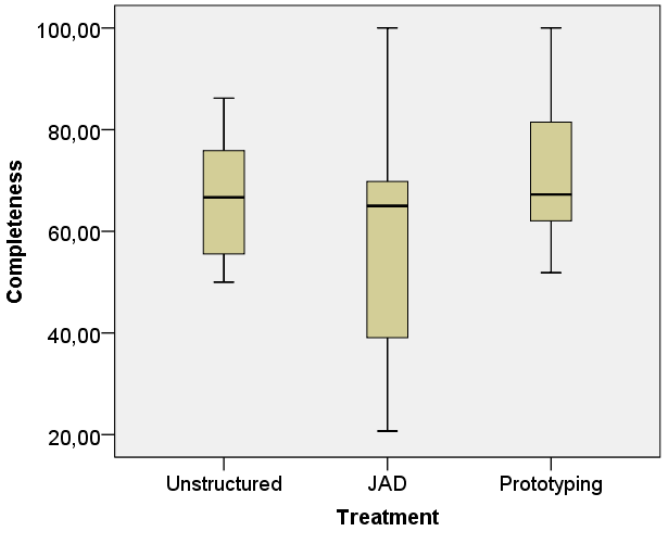

Fig 7. Box-and-whisker plot for Completeness

Table 8 shows the results of the statistical analysis for the metric of completeness adopted. P-value yields that there are not significant differences among the methods (p-value > 0.05), thus we conclude that we cannot reject *H_{04}: The completeness of requirements is similar among JAD, Paper Prototyping and Unstructured Interviews.* This means that **the level of completeness of the requirements elicitation document is the same, independently of the method applied.** Note that the power of the statistical test is low, which means that it is possible that we are not rejecting the null hypothesis due to the low number of experimental units.

Table 8. Statistical analysis for the metric for Completeness

	Mean	Std. Deviation	p-value	Effect size
Completeness	65.28	19.30	0.17	-

Response variable **Quality** has only one metric calculated as the percentage of requirements (independently whether they are or not in instructors' solution) that are specified correctly per the total amount of requirements in instructors' solution. Fig 8 shows the box-and-whisker plot for this metric. Medians are very similar among the three methods, which means that there are not important differences in Quality. The method with the best median, first quartile and third quartile is Paper Prototyping. This means that Paper Prototyping is the method that allows to get the closest solution to the instructor's requirements document.

Table 9 shows the results of the statistical analysis. According to the p-value, we do not identify significant differences among the methods (p-value>0.05). So, we cannot reject *H_{06}: The quality of requirements is similar among JAD, Paper Prototyping and Unstructured Interviews.* This means that the percentage of requirements correctly elicited by the subjects is similar, independently of the method applied. This lack of significance might be due to the low statistical power.

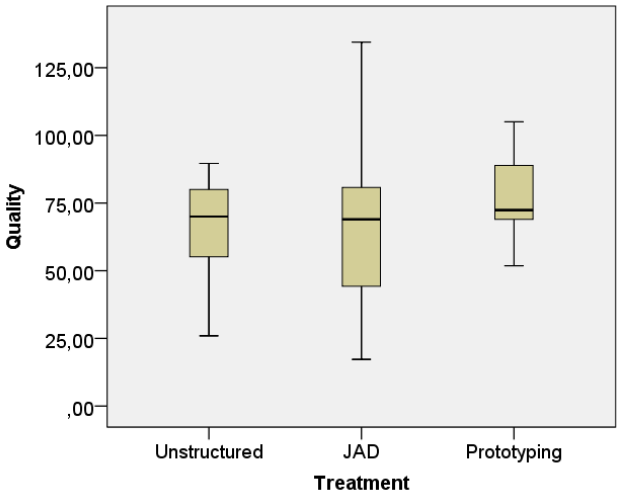

Fig 8. Box-and-whisker plot for quality

Table 9. Statistical analysis for Quality

	Mean	Std. Deviation	p-value	Effect size
Quality	68.82	25.76	0.59	-

Response variable **Performance** is calculated through three metrics: number of questions asked to the client (Performance_num_questions), number of relevant questions asked to the client (Performance_num_relevant_questions), efficiency measured as the ratio Quality/OverallTime (Performance_efficiency). Fig 9.a shows the box-and-whisker plot for the number of questions during the interview. Medians of the three methods are very similar, so we do not appreciate differences among them. The median, first and third quartile are slightly higher for JAD, which means that this method leads to asking more questions than the others. In Fig 9.b we have the box-and-whisker plot for the number of relevant questions during the interview. Results are the same as the number of questions previously commented (Fig 9.a). Fig 10 shows the box-and-whisker for efficiency. The median of JAD is considerably different from the other two ones; in particular, JAD obtains the worst values for efficiency. This means that, compared to the other two methods, JAD requires too much time for the quality that it gets in the elaborated requirements elicitation document. The method with the best median and the best third quartile is Paper Prototyping, which means that it is the most efficient among the three methods.

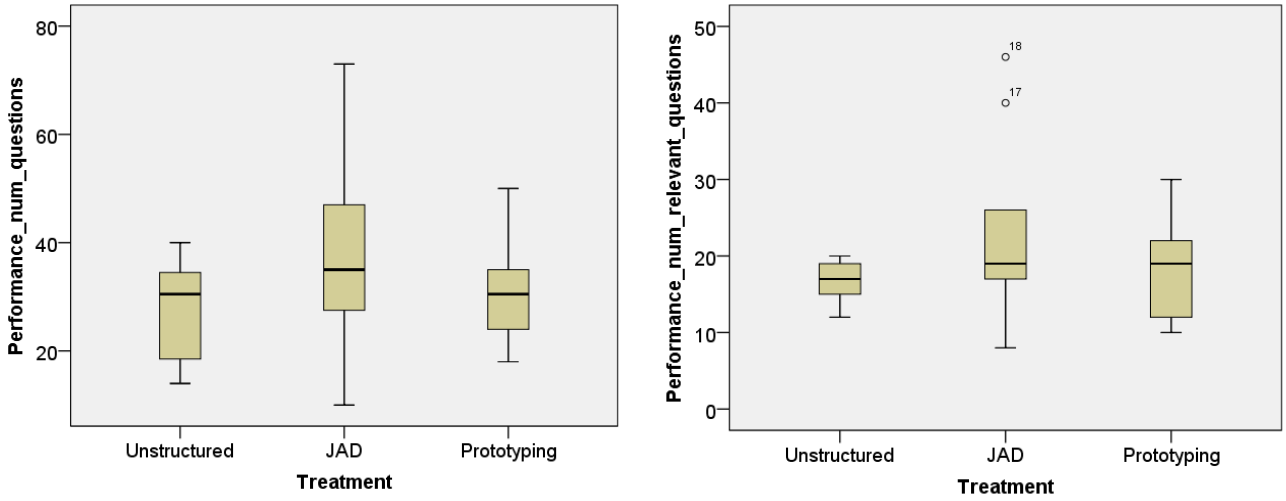

Fig 9. Box-and-whisker plot for (a) number of questions and (b) number of relevant questions

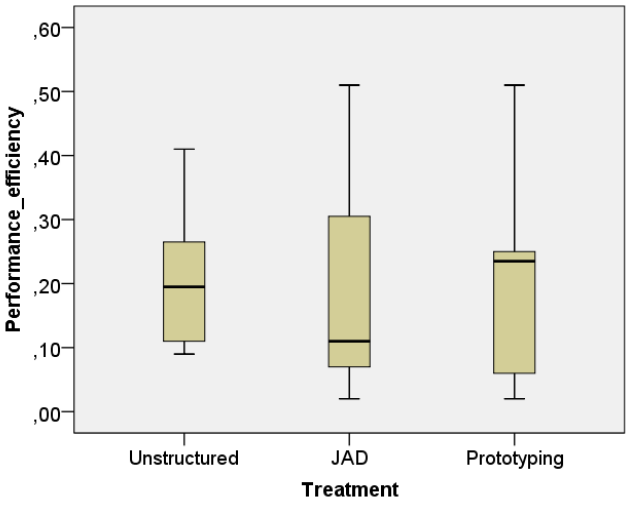

Fig 10. Box-and-whisker plot for efficiency

Table 10 shows the results of the statistical analysis for the metrics of Performance. According to p-values, we did not find significant differences among the methods (p-value>0.05). So, we conclude that we cannot reject *H_{05}: The performance to elicit requirements is similar among JAD, Paper Prototyping and Unstructured Interviews.* This means that **from the point of view of resources** (number of questions and time) **the three methods accomplish the requirements elicitation process in a similar way**. The lack of significance might be due to the low power in the same way as for Completeness.

Table 10. Statistical analysis for metrics of performance

	Mean	Std. Deviation	p-value	Effect size
Performance_num_questions	32.76	13.92	0.55	-
Performance_num_relevant_questions	19.65	8.90	0.63	-
Performance_efficiency	0.20	0.15	0.82	-

6 Discussion

This section compares Unstructured Interviews, JAD and Paper Prototyping according to the results analyzed in the previous section. For each method, we analyze the pros and cons, as well as under which context each one of these methods is more suitable to apply.

Unstructured Interviews is the most flexible method. Subjects could organize the meeting as they preferred, without roles or artifacts that help in the requirements elicitation process. One of the advantages of using this method is that it requires the lowest time to report the requirements in the requirements elicitation document. This difference in time to report is significant according to the statistical analysis. This result can be justified through one of the disadvantages, this method yields the lowest number of functional requirements and the lowest number of the addition of functional and non-functional requirements. These differences are significant in the case of the addition of functional and non-functional requirements. Since Unstructured Interviews allows to identify fewer requirements, the time to report the requirements is the lowest. Another advantage of Unstructured Interviews is that it is the method that needs less time to perform the interviews with the client. The lack of a specific protocol involves that interviews are more direct. Regarding the overall time of the elicitation requirements process, Unstructured Interviews also yields the best value, it is the shortest one. Note that the time to prepare the interviews is considerably lower than the other two methods. Subjects did not have to study how to organize the interviews or how to deal with artifacts such us, prototypes. Unstructured Interviews is also the method that asks fewer questions during the interview with the client. Moreover, there are many questions that are not relevant, which involves that time in the interviews could be optimized, reducing the overall time even more. The lack of a protocol to prepare the interviews beforehand might justify the high number of non-relevant questions.

Another advantage of Unstructured Interviews is that even though the number of requirements identified is lower than the other two methods, the level of completeness is not so low. This means that both JAD and Paper Prototyping are identifying more requirements than Unstructured Interviews even though many of these requirements are not part of the instructors' solution (actually they are not requirements).

Apart from the disadvantage regarding the low number of requirements described previously, Unstructured Interviews has other concerns. It is the method with the highest value for diversity, which means that it has the highest overlapping among the three methods. Note that this difference in diversity is significant according to the statistical test. This result implies that there are several requirements that are repeated in the requirements elicitation document. Considering that Unstructured Interviews is the method with the smallest number of requirements identified and that several of the identified requirements are overlapped, this method yields the worst values regarding the number of relevant requirements.

As a conclusion, we state that Unstructured Interviews is a method easy to use, it does not require to read several manuals with protocols or artifacts. It is not the best method to get all the requirements, but the completeness of the identified requirements is similar to JAD and Paper Prototyping. Unstructured Interviews is the fastest method to elicit requirements, the time to learn is very short and interviews are done quickly. So, this method is suitable for contexts where timesaving is important and we do not need a high level of quality. The lack of a specific protocol makes that there is an important number of irrelevant questions asked for the extraction of requirements. Moreover, there are many requirements that are overlapped. The overlapping together with non-relevant questions lead to think that time for Unstructured Interviews could be optimized even more than the current numbers obtained in the experiments. The effectiveness of this method could be high at the very beginning of the process, in groups of developers with little experience and in the context of not large problems to solve.

JAD is the method with the clearest distribution of duties: every member of the team knows her/his role in the interview and what to do. One of the identified advantages of JAD is that it is the method that elicits more non-functional requirements. This could be because the persons that participate in the interview have more consciousness of the need for non-functional requirements. Regarding the functional requirements, their amount is significantly higher in JAD than in Unstructured Interviews, with similar quality both in JAD and in Unstructured Interviews. Another advantage of JAD is that this is the method with the best diversity, and this difference is significant according to the statistical test. The number of overlapping in the requirements elicitation is very close to 0. This means that JAD allows focusing on the interview requirement per requirement more independently than the other two methods. Another advantage observed in JAD is that it is the method with more questions asked to the client, and the method with more relevant questions. The existence of a specific role for asking might justify the higher number of relevant questions. The subject with the role of asking could prepare the questions in advance, so most of the questions were relevant because they were prepared consciously. In the other two methods without roles, questions were not so prepared, so the improvisation was probably the cause for the higher number of non-relevant questions during the interviews.

The highest number of non-functional requirements identified and the highest number of questions during the interviews involve that the time spent is JAD is higher than with the other two methods. All the measured times are higher in JAD. The time needed to prepare the interview is higher since all the members of the team must meet before the interviews with the client in order to distribute the roles and to coordinate the tasks. This previous meeting is not so important in the case of Unstructured Interviews and Paper Prototyping, so times are lower. Time spent in interviews with the client is higher than Unstructured Interviews but very similar to Paper Prototyping. The time that all roles need to participate in the interview with JAD is similar to the time that Paper Prototyping needs to validate the prototypes with the client in the interviews. Time to report elicited requirements is the highest with JAD. This difference is significant according to the statistical test. This is a curious result since, at first sight, the fact of a good structure within the team should lead to report requirements quickly. Maybe, the fact that there is no specific role to write the requirements document after the interview could justify this high time. In the end, the addition of the whole process with JAD takes more time than Unstructured Interviews and Paper Prototyping. Note that apart from the time spent within the interview, roles spent time to coordinate each other and to prepare the role-playing in the interview, which increases the amount of time of the whole process.

Another disadvantage of JAD is that the level of completeness is slightly lower than the other two methods. This involves that even though JAD yields the higher number of relevant questions, there are still requirements that are missing in the elicitation process. Regarding performance, JAD is the method with the lowest level of efficiency. This means that the amount of time spent in the whole process does not involve a better requirements elicitation document compared with the other two methods. In other words, the quality of JAD is very similar to the quality of Unstructured Interviews and Paper Prototyping but the required time is higher.

As a conclusion, we state that JAD is a method that requires a lot of effort before the interview. Participants had to coordinate each other, distribute the roles and prepare the interview. As benefits, the precision of the asked questions is high, which means that the overlapping of requirements is close to 0. So, JAD could be used more suitable in a context where there is a lack of coordination among all the members of the team and there is time to prepare the interview in advanced. JAD is not recommended in contexts where the time spent must be short. The effectiveness of this method could be high in groups of developers with a long experience in the role played in the team. The division of roles makes it easier to tackle complex problems, since it allows the specialization of the members in a single task.

Paper Prototyping is a method that helps the elicitation process through prototypes. One of the advantages of Paper Prototyping is that this method yields the highest number of identified functional requirements, and this difference is significant according to the statistical test. The use of prototypes helps in the identification of functional requirements.

However, these prototypes do not seem to help in the elicitation of non-functional requirements. Paper Prototyping yields the worst number of identified non-functional requirements and the total number of requirements is similar to JAD. In general, time spent to apply Paper Prototyping is similar to Unstructured Interviews. Time to prepare the interviews is shorter than with JAD. At first sight, Paper Prototyping should involve more time than JAD or Unstructured Interviews since, apart from the requirements elicitation document, Paper Prototyping needs to create the prototypes. However, time spent in the construction of the prototypes involves a reduction of the time to elicit the requirements since interviews are shorter. Moreover, both JAD and Paper Prototyping involve the study of the method, but, in the case of Paper Prototyping, the method seems to be easier to learn. Time spent within the interview is similar to the time with JAD. This means that the use of prototypes does not require more time than Unstructured Interviews or JAD in the interviews. Time spent to report the requirements in a document is similar to Unstructured Interviews and shorter than JAD. This difference is significant according to the statistical test. This might be because, at the end of the interviews, prototypes can be considered as a summary of the functional requirements. This way, translating the prototypes to a requirements elicitation document textually is easier than writing the document from scratch in the case of JAD or Unstructured Interviews. The overall time of the whole process with Paper Prototyping is similar to Unstructured Interviews and lower than JAD.

Another advantage of Paper Prototyping compared to Unstructured Interviews is the good values for diversity. Even though JAD yields better results than Paper Prototyping, the overlapping with Paper Prototyping is significantly better than with Unstructured Interviews, and these differences are significant according to the statistical tests. This means that the prototypes help split the different requirements. Comparing Paper Prototyping versus JAD, we state that the use of roles with specific tasks is the best method to avoid overlapping among requirements. Regarding completeness, Paper Prototyping yields the best value, even though this difference is very small. This leads to think that all three methods are similar to identify the different requirements that should appear in the requirements document. Another advantage of Paper Prototyping is that this method yields the least number of non-relevant questions. This is related with the short time spent in the process. Questions in interviews are more accurate, which reduces the required time in the interview. Regarding efficiency, it is quite similar to Unstructured Interviews and higher compared to JAD. This means that the time invested in building prototypes is time that improves the correctness of the requirements list. The last advantage we have found is that Paper Prototyping yields the best value for quality. This means that the number of requirements that appear in the requirements list is very similar to the instructors' solution.

Regarding the disadvantages of Paper Prototyping, we have identified that the use of prototypes does not help in the elicitation of non-functional requirements. Paper Prototyping yields the lowest number of non-functional requirements, close to the number reached through Unstructured Interviews. This might be because the use of prototypes focuses the interviews on functional requirements, pushing non-functional requirements in the background. Regarding diversity, Paper Prototyping yields better results than JAD, however, the overlapping is lower than with Unstructured Interviews. This result may be due to the fact that the same prototype may gather different requirements and subjects used to write one requirement per prototype. This means that requirements that appear in more than one prototype are overlapped.

As a conclusion, we state that Paper Prototyping is the method that yields the best quality and completeness in the elaboration of the requirements elicitation document. Paper Prototyping is specifically suitable for functional requirements; if we aim to focus on non-functional requirements, this is not the best choice. Paper Prototyping is not so fast as Unstructured Interviews, but it is the most efficient, since time spent in the elaboration of the prototypes involves more quality in the requirements elicitation process. So, Paper Prototyping is recommendable for developers that need to extract requirements the most accurate as possible, and where there are no time restrictions in the short term. The effectiveness of this method could be high when the elicitation process is in an advanced stage, when the meeting can be focused on prototypes. The team does not require a long experience to apply the method, and the use of prototypes helps in the elicitation of large systems, since they help to guide the interview.

7 Conclusions

This paper presents the design and the conduction of an experiment to compare three methods, based on interviews, to elicit requirements in a team of developers: Unstructured Interviews, JAD and Paper Prototyping. Unstructured Interviews consists in performing the interviews with no specific protocol, roles distribution or artifacts. JAD is a method based on role-playing distribution where each member of the team has a specific task within the team. Paper Prototyping consists in using prototypes to guide the interview. During the interview, developers discuss with the client the prototypes they built previously. The experiment to compare these three methods is a between-subjects design replicated 4 times during 3 years. The replications were conducted in the subject of Software Engineering in the Degree in Computer Engineering and the Degree in Telematics Engineering at Universitat de València (Spain). We used three different problems (one per year) and we recruited 167 subjects divided into 32 experimental units (since subjects worked in teams).

Results of the experiment show that there are some significant differences among the three methods. Paper Prototyping yields the highest number of elicited functional requirements. JAD is the method that requires more time to report the requirements and also the method with the least overlapping among requirements. Unstructured Interviews is the method that requires the least time to report the requirements. Apart from significant differences, we found also other differences analyzing descriptive statistics. In summary, Unstructured Interviews is the fastest method, but with the poorest diversity and performance. JAD yields the highest time, identifies the highest number of non-functional requirements and gets the best diversity, but its performance and quality are low. Paper Prototyping requires low time to elicit requirements and the performance and quality are the best, but it is not suitable to identify non-functional requirements.

It is important to note that our experiment suffers several limitations. One of these limitations is the subjects' profile. Subjects were students with no experience in a real company. The lack of experience provides the advantage that subjects apply the method without hints, but this also involves that results cannot be generalized to industrial contexts. Another limitation is that even though we analyzed 4 replications during 3 years, the number of experimental units is not so high. Our experiment suffers the lack of statistical power and we performed the analysis through non-parametric tests. This involves that some null hypotheses were not rejected due to the lack of possible significant differences but due to the lack of power. Another limitation of our experiment is that we have focused our study on the elaboration of the requirements elicitation document. This is a small piece in the whole software development process. How each method works after requirements are elicited is out of the scope of our study. It is true that the analyzed methods are for requirements elicitation, but the process of elicitation usually does not finish in a few interviews, but spreads through different steps of a software development process. The last limitation of our experiment is the type of metrics used in the analysis. Most of the metrics are based on the requirements elicitation document, which is a textual artifact. Usually, textual artifacts lead to misunderstandings and subjective ideas that are difficult to measure.

Apart from the limitations, the experiment has several strengths. Data to analyze the results have been extracted from 4 replications during 3 years with different subjects, which means that we analyzed the response variables through different contexts with the same design. The use of different contexts helps in the generalization of the results. Another advantage is that even though we did not find significant differences for all the response variables, we identified some trends looking at the descriptive statistics. These tendencies might be checked with more experimental units in future replications. Conclusions extracted from our family of replications are applicable to teams with several members that work in coordination to elicit requirements, which could be a wide number of potential users.

As future work, we plan to study how requirements elicitation methods may affect other development steps (beyond the elaboration of the requirements elicitation document). We would like to study how the method affects parts of the analysis and design of the system. Note that how requirements are elicited drives the whole software development process and some issues in the elicitation step can only be seen in later steps. Moreover, in order to increase the experimental units and enhance the statistical power, we plan to replicate the same experiment in other universities with different instructors. This will help us apply parametric tests to analyze the data. As conclusions of all the replications, we plan to define a new method to elicit requirements that gathers all the benefits of the studied methods.

Acknowledgments

This project has been developed with the financial support of the Spanish Ministry of Science and Innovation through project DataME (ref: TIN2016-80811-P) and co-financed with ERDF.

8 References

1. Pohl, K.: Requirements Engineering: Fundamentals, Principles, and Techniques. Springer Publishing Company, Incorporated (2010)

2. Zowghi, D., Coulin, C.: Requirements Elicitation: A Survey of Techniques, Approaches, and Tools. In: Aurum, A., Wohlin, C. (eds.): Engineering and Managing Software Requirements. Springer Berlin Heidelberg, Berlin, Heidelberg (2005) 19-46

3. Friedrich, W., Van der Poll, J.: Towards a Methodology to Elicit Tacit Domain Knowledge from Users. Interdisciplinary Journal of Information, Knowledge, and Management **2** (2007)

4. Carmel, E., Whitaker, R.D., George, J.F.: PD and joint application design: a transatlantic comparison. Commun. ACM **36** (1993) 40-48

5. Sommerville, I.: Software Engineering. Pearson (2010)

6. Neill, C.J., Laplante, P.A.: Requirements engineering: the state of the practice. IEEE Software **20** (2003) 40-45

7. Hofmann, H.F., Lehner, F.: Requirements engineering as a success factor in software projects. IEEE Software **18** (2001) 58-66

8. Basili, V.R., Lanubile, F.: Building Knowledge through Families of Experiments. IEEE Transaction on Software Engineering **25** (1999) 456-473

9. IEEE: IEEE standard 830-1998. (1998)

10. Dieste, O., Juristo, N.: Systematic review and aggregation of empirical studies on elicitation techniques. IEEE Transactions on Software

Engineering **37** (2011) 283-304
11. Sharma, P., Dhir, S.: Functional & non-functional requirement elicitation and risk assessment for agile processes. Vol. 9 (2016) 9005-9010
12. Rodriguez, G., Wong, L., Mauricio, D.: A systematic literature review about software requirements elicitation. Vol. 12 (2017) 296-317
13. Carrizo, D., Dieste, O., Juristo, N.: Systematizing requirements elicitation technique selection. Information and Software Technology **56** (2014) 644-669
14. Spoletini, P., Ferrari, A., Bano, M., Zowghi, D., Gnesi, S.: Interview Review: An Empirical Study on Detecting Ambiguities in Requirements Elicitation Interviews. Springer International Publishing, Cham (2018) 101-118
15. Ferrari, A., Spoletini, P., Donati, B., Zowghi, D., Gnesi, S.: Interview Review: Detecting Latent Ambiguities to Improve the Requirements Elicitation Process. 2017 IEEE 25th International Requirements Engineering Conference (RE) (2017) 400-405
16. Burnay, C., Snoeck, M.: Trust in requirements elicitation: how does it build, and why does it matter to requirements engineers? : Proceedings of the Symposium on Applied Computing. ACM, Marrakech, Morocco (2017) 1094-1100
17. Ambreen, T., Ikram, N., Usman, M., Niazi, M.: Empirical research in requirements engineering: trends and opportunities. Requirements Engineering (2016) 1-33
18. Kumari, S.N., Pillai, A.S.: Requirements elicitation issues and project performance: A test of a contingency model. 2015 Science and Information Conference (SAI) (2015) 889-896
19. Sethia, N.K., Pillai, A.S.: The Effects of Requirements Elicitation Issues on Software Project Performance: An Empirical Analysis. In: Salinesi, C., van de Weerd, I. (eds.): Requirements Engineering: Foundation for Software Quality. Springer International Publishing, Cham (2014) 285-300
20. Farinha, C., Mira da Silva, M.: Requirements Elicitation with Web-Based Focus Groups (2011)
21. Besrour, S., Rahim, L.B.A., Dominic, P.D.D.: Assessment and evaluation of requirements elicitation techniques using analysis determination requirements framework. 2014 International Conference on Computer and Information Sciences (ICCOINS) (2014) 1-6
22. Brill, O., Schneider, K., Knauss, E.: Videos vs. Use Cases: Can Videos Capture More Requirements under Time Pressure? Springer Berlin Heidelberg, Berlin, Heidelberg (2010) 30-44
23. Nascimento, E., Silva, W., Conte, T., Steinmacher, I., Massollar, J., Travassos, G.H.: Is a Picture worth a Thousand Words?: A Comparative Analysis of Using Textual and Graphical Approaches to Specify Use Cases. Proceedings of the 30th Brazilian Symposium on Software Engineering. ACM, Maringá, Brazil (2016) 93-102
24. Ibriwesh, I., Dollmat, K.S., Ho, S.-B., Chai, I., Tan, C.-H.: A Controlled Experiment on Requirements Elicitation in Electronic Markets. Proceedings of the 8th International Conference on E-Education, E-Business, E-Management and E-Learning. ACM, Kuala Lumpur, Malaysia (2017) 56-60
25. Hadar, I., Soffer, P., Kenzi, K.: The role of domain knowledge in requirements elicitation via interviews: an exploratory study. Requirements Engineering **19** (2014) 143-159
26. Lloyd, W.J., Rosson, M.B., Arthur, J.D.: Effectiveness of elicitation techniques in distributed requirements engineering. Proceedings IEEE Joint International Conference on Requirements Engineering (2002) 311-318
27. Mahmood, S., Ajila, S.A.: Software Requirements Elicitation -- A Controlled Experiment to Measure the Impact of a Native Natural Language. 2013 IEEE 37th Annual Computer Software and Applications Conference (2013) 437-442
28. Aranda, A.M., Dieste, O., Juristo, N.: Effect of Domain Knowledge on Elicitation Effectiveness: An Internally Replicated Controlled Experiment. IEEE Transactions on Software Engineering **42** (2016) 427-451
29. Riaz, M., King, J., Slankas, J., Williams, L., Massacci, F., Quesada-López, C., Jenkins, M.: Identifying the implied: Findings from three differentiated replications on the use of security requirements templates. Empirical software engineering **22** (2017) 2127-2178
30. Morales-Ramirez, I., Kifetew, F.M., Perini, A.: Speech-acts based analysis for requirements discovery from online discussions. Information Systems (2018)
31. Adam, S., Schmid, K.: Effective requirements elicitation in product line application engineering: an experiment. Proceedings of the 19th international conference on Requirements Engineering: Foundation for Software Quality. Springer-Verlag, Essen, Germany (2013) 362-378
32. Gralha, C.: Evaluation of Requirements Models. 2016 IEEE 24th International Requirements Engineering Conference (RE) (2016) 432-437
33. Gacitua, R., Sawyer, P., Gervasi, V.: On the Effectiveness of Abstraction Identification in Requirements Engineering. 2010 18th IEEE International Requirements Engineering Conference (2010) 5-14
34. Vitharana, P., Zahedi, M.F., Jain, H.K.: Enhancing analysts' mental models for improving requirements elicitation: A two-stage theoretical framework and empirical results. Journal of the Association of Information Systems **17** (2016) 804-840
35. Yamanaka, T., Komiya, S.: A method to navigate interview-driven software requirements elicitation work: an efficacy evaluation of the method from the viewpoint of efficiency. Proceedings of the 2010 international conference on Applied computing conference. World Scientific and Engineering Academy and Society (WSEAS), Timisoara, Romania (2010) 54-60
36. Browne, G.J., Rogich, M.B.: An Empirical Investigation of User Requirements Elicitation: Comparing the Effectiveness of Prompting Techniques. J. Manage. Inf. Syst. **17** (2001) 223-249
37. Ghanbari, H., Similä, J., Markkula, J.: Utilizing online serious games to facilitate distributed requirements elicitation. Journal of Systems and Software **109** (2015) 32-49
38. Lombriser, P., Dalpiaz, F., Lucassen, G., Brinkkemper, S.: Gamified Requirements Engineering: Model and Experimentation. Springer

International Publishing, Cham (2016) 171-187
39.Leffingwell, D.: Agile Software Requirements: Lean Requirements Practices for Teams, Programs, and the Enterprise. Addison-Wesley Professional (2010)
40.Vijayan, J., Raju, G.: Requirements Elicitation Using Paper Prototype. In: Kim, T.-h., Kim, H.-K., Khan, M.K., Kiumi, A., Fang, W.-c., Ślęzak, D. (eds.): Advances in Software Engineering. Springer Berlin Heidelberg, Berlin, Heidelberg (2010) 30-37
41.Mannio, M., Nikula, U.: Requirements elicitation using a combination of prototypes and scenarios. WER (2001) 283-296
42.Warfel, T.Z.: Prototyping. Rosenfeld Media (2009)
43.Balsamiq: https://balsamiq.com/wireframes/desktop/.
44.Singer, J., Vinson, N.G.: Ethical issues in empirical studies of software engineering. IEEE Transactions on Software Engineering **28** (2002) 1171-1180
45.Tiwari, S., Gupta, A.: A systematic literature review of use case specifications research. Information and Software Technology **67** (2015) 128-158
46.IEEE: Systems and software engineering -- Vocabulary. ISO/IEC/IEEE 24765:2010(E) (2010) 1-418
47.Juristo, N., Moreno, A.: Basics of Software Engineering Experimentation. Springer (2001)
48.Chung, L., Nixon, B., Yu, E., Mylopoulos, J.: Non-Functional Requirements in Software Engineering. Kluwer Academic Publishing, London (2000)
49.Wohlin, C., Runeson, P., Höst, M., Ohlsson, M.C., Regnell, B., Wesslén, A.: Experimentation in Software Engineering: An Introduction. Springer (2012)
50.Pajula, J., Tohka, J.: How Many Is Enough? Effect of Sample Size in Inter-Subject Correlation Analysis of fMRI. Computational intelligence and neuroscience **2016** (2016) 2094601-2094601
51.Faul, F., Erdfelder, E., Lang, A.-G., Buchner, A.: G*Power 3: A flexible statistical power analysis program for the social, behavioral, and biomedical sciences. Behavior Research Methods **39** (2007) 175-191
52.Dalgaard, P.: Analysis of variance and the Kruskal–Wallis test. Introductory Statistics with R. Springer New York, New York, NY (2008) 127-143
53.Zuur, A.F., Ieno, E.N., Elphick, C.S.: A protocol for data exploration to avoid common statistical problems. Methods in ecology and evolution **1** (2010) 3-14