The Influence of Requirements in Software Model Development in an Industrial Environment

Jorge Echeverría*, Francisca Pérez*, Jose Ignacio Panach†, Carlos Cetina* and Óscar Pastor‡
*Universidad San Jorge
Autovía A-23 Zaragoza-Huesca Km.299, Zaragoza, Spain
Email: {jecheverria, mfperez, ccetina}@usj.es
†Universitat de València
Avenida de la Universidad, s/n, 46100 Burjassot, Valencia, Spain
Email: joigpana@uv.es
‡Universitat Politècnica de València
Camino de Vera, s/n, 46022 , Valencia, Spain
Email: opastor@pros.upv.es

***Abstract*—Textual description of requirements is a specification technique that is widely used in industry, where time is key for success. How requirements are specified textually greatly depends on human factors. In order to study how requirements processing is affected by the level of detail in textual descriptions, this paper compares enriched textual requirements specifications with non-enriched ones. To do this, we have conducted an experiment in industry with 19 engineers of CAF (Construcciones y Auxiliares de Ferrocarril), which is a supplier of railway solutions. The experiment is a crossover design that analyzes efficiency, effectiveness, and perceived difficulty starting from a written specification of requirements that subjects must process in order to build software models. The results show that effectiveness and efficiency for enriched requirements are better, while non-enriched requirements are slightly more difficult to deal with. Therefore, even though enriched requirements require more time to be specified, the results are more successfully when using them.**

***Index Terms*—Software Requirements, Model-Based Software Development, Controlled Experiment**

I. Introduction

Requirements elicitation is a process that greatly depends on human factors. Cappel [6] concluded that non-technical skills such as oral and written communication, problem solving, and the ability to learn all apply to the requirements elicitation process. How subjective human factors can affect the elicitation process is usually dealt with in the field of psychology and is out of the scope of this paper.

There are many techniques for eliciting requirements, such as prototypes, structured interviews, task observation, surveys, brainstorming, nominal group techniques, and focus groups, among many others. There are also theoretical works that have studied which techniques are better depending on the context [7]. To establish which requirement elicitation technique is better from a practical point of view is still a challenging study for the software engineering community.

Of all of the existing techniques, this paper focuses on studying the textual description technique, where the analyst writes down a textual specification that describes all of the features of the system that are to be developed. The provided description acts as input for the next steps of the development process. A textual description is quite subjective since the degree of detail of the description can vary greatly depending on the subject.

The goal of this paper is to analyze which type of requirement description is better: a non-enriched requirement or an enriched requirement. A non-enriched requirement description contains only natural language. On the other hand, the enriched requirement description contains natural language that is enhanced with property-value pairs of elements that are included in the description.

This paper proposes an experiment design that is based on a crossover design. The subjects start from an already existing textual description of requirements and must process them to build software models. The recruitment of subjects was carried out with software engineers of our industrial partner company, CAF (http://www.caf.net/en) at two of their headquarters. One of the headquarters is located in Zaragoza (Spain), where we recruited six subjects; and the other one is located in Beasain (Spain), where we recruited thirteen subjects. CAF is a worldwide supplier of railway solutions. Their trains can be found all over the world in different forms (regular trains, subway, light rail, monorail, etc.).

The experiment was conducted in terms of three variables: effectiveness (ratio of errors), efficiency (time spent), and perceived difficulty (subjective feelings of the software engineer). The results show that there are significant differences between enriched and non-enriched requirements for these three variables, demonstrating that effectiveness and efficiency for enriched requirements have better values, while non-enriched requirements are slightly more difficult to deal with than enriched ones.

The paper is structured as follows: Section 2 explains the background of enriched and non-enriched requirements. Section 3 describes the design of the experiment. Section 4 shows the statistical results. Section 5 discusses the results. Section 6 deals with the threats to validity. Section 7 analyzes other works that have compared requirements elicitation techniques. Finally, Section 8 presents some relevant conclusions.

II. Background

Requirements engineering is treated as one of the most important parts of the software development process. The requirements process has been identified as being a crucial factor for the success of a project. This is particularly valid in large one-of-a-kind projects that require significant effort in the initial product-specification stages, where a large proportion of the product cost is committed. In fact, when developing products of high complexity in industries such as aerospace, it is widely acknowledged that almost 60% of the product cost is allocated during the first 5% of the product development cycle [23].

In general, requirements are expressed using free natural language text in a large number of software projects, and the railway domain is no exception [26]. Natural language is used to specify requirements due to its high degree of understandability among all of the stakeholders in industrial projects [12].

When we focused our analysis on CAF, we identified that the company works with mainly two types of textual requirement descriptions (non-enriched and enriched):

- **The non-enriched requirement** description contains only natural language. The following statement is an example of a non-enriched requirement: *The PLC will inhibit the circuit breaker shutdown whenever there is inhibition from ACR management, when there is outside power, or when permission isn't granted from any UCU.*
- **The enriched requirement** description contains natural language that is enhanced with property values of model elements that are included in the description. An example of this kind of requirement is: *The PLC will inhibit the circuit breaker shutdown (AT_INHIB_HSCB=1) whenever there is inhibition from ACR management (SC_INHIB_HSCB=1), when there is outside power (CV_POWER_OUT=1, AT_POWER_OUT=1) or when permission isn't granted from any UCU (UCU_Cx_ControlW.b_HSCBPerm=0).* Fig. 1 shows the description of an enriched requirement with the property-value pairs. These property-value pairs are denoted with a dotted line.

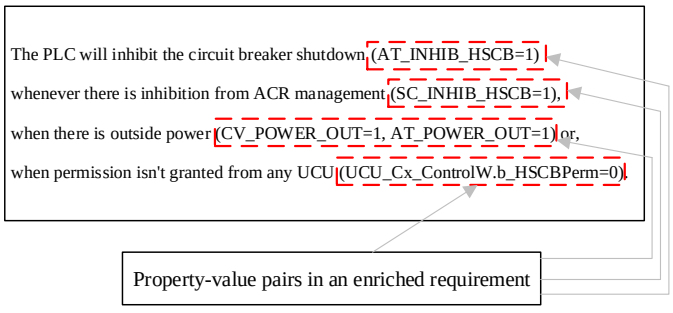

Fig. 1. Enriched requirement with property-value pairs

Both enriched or non-enriched requirements are the input to build software models that describe the functionality of their trains. The modeling language is the Train Control and Management Language (TCML), which is a domain specific language that has both the expressiveness required to describe the interaction between the main elements in a train and the expressiveness required to specify non-functional aspects that are related to regulation.

Our goal is to analyze which of the two types of textual descriptions of requirement is better for building software models in an industrial environment. For this reason, we conducted an experiment at the CAF company, where subjects must build a software model according to a requirement description. Our aim is to compare the results using enriched requirements versus non-enriched requirements.

III. Experiment Design

A. Objective

The goal of our research is to analyze which of the two types of textual description is better as a starting point to build software models in an industrial environment. Following the Wohlin et al. guidelines [30], the goal of our study was to:

Analyze the performance of engineers when they process requirements;

For the purpose of filling in the gap in empirical evaluation on this topic;

With respect to the different enrichment levels of requirements;

From the viewpoint of software engineers;

In the context of a train manufacturing company.

The measures used in our research to achieve the above goal are effectiveness, efficiency, and perceived difficulty. Effectiveness and efficiency are widely accepted to measure the performance of software engineers [9], [17]. In addition, perceived difficulty is one of the most widely applied theoretical models when analyzing user acceptance in the software engineering research community [10], [24]. We seek to answer the following three research questions using these measures:

RQ1 Are there differences when different enrichment levels of requirements are used regarding effectiveness in building software models?

RQ2 Are there differences when different enrichment levels of requirements are used regarding efficiency in building software models?

RQ3 Is the perceived difficulty different when software engineers use different enrichment levels of requirements to build software models?

To answer these research questions, we have formulated the following null hypotheses:

- $\mathbf{H_{01}}$: There is no difference in the effectiveness of the performance of software engineers when working with requirements with different enrichment levels.
- $\mathbf{H_{02}}$: There is no difference in the efficiency of the performance of software engineers when working with requirements with different enrichment levels.
- $\mathbf{H_{03}}$: There is no difference in the perceived difficulty of software engineers when working with requirements with different enrichment levels.

B. Participants

The subjects were 19 software engineers from the CAF company. These engineers are experts in developing software and requirements. In their daily work, these engineers develop software from both enriched and non-enriched requirements. Six software engineers work at the company headquarters in Zaragoza (Spain), and thirteen software engineers work at the company headquarters in Beasain (Spain). They have spent from 1 to 15 years working as software engineers (a mean of 6.65 years). They claimed that they spent an average of 3.68 hours per day developing software. In addition, the software engineers stated that they spent a mean of 3.36 hours per day developing requirements.

Apart from the subjects, an instructor, two observers, and one domain expert were also involved. The instructor provided information about performing the exercises, clarified doubts during the experiment, and managed the focus groups. The observers took notes for further analyses. Finally, the domain expert stated the requirements, designed the correction, corrected the exercises, and solved doubts about the requirement statements for the subjects during the experiment. The domain expert was not involved in this paper.

C. Defining Variables

1) Independent Variables: We conducted a single factor experiment where the independent variable is the enrichment level of the requirement, which is a nominal variable with two values: enriched requirements and non-enriched requirements. These levels are explained in section II.

2) Dependent Variables: In our experiment, the software engineers had to perform four exercises, where every exercise contained a requirement description. The outcome of each exercise was a software model that used concepts that were known by the subjects. During the transformation from the requirements specification to software model, we measured the effectiveness, efficiency, and perceived difficulty. The dependent variables are defined as follows:

- Effectiveness is defined as the percentage of an exercise performed correctly by the software engineer. The exercises are decomposed by a domain expert into a set of steps, and each step has a weighted percentage with respect to the whole exercise.
- Efficiency is the ratio between the effectiveness and the time spent (in minutes) to perform the exercise.
- The perceived difficulty is the perception that a software engineer has of the complexity of an exercise. It is measured using a Likert scale. The software engineers must fill in a value for the perceived difficulty for each exercise.

D. Instruments

1) Demographic Questionnaire: This includes questions to identify the profile of each subject. The information requested in the questionnaire is: their education level, the length of time working in their current department (in years), their age, their gender, the time spent per day developing software, and the time spent per day developing requirements.

2) Task sheet: A task sheet for each requirement was given to the subjects. Every task sheet contains a requirement description, a Likert scale, a text area, and two text fields. The requirement description is the requirement to be processed. The subjects also had to fill in the value within the Likert scale ranging from 1=very easy to 7=very difficult. This value is used to calculate the perceived difficulty. In the text area, the subjects could write down any opinions they had about the process of converting requirements into software models. Finally, the subjects had to write down both the time they started the exercise to build software models from requirements and the time they finished. These times provide the calculation for efficiency.

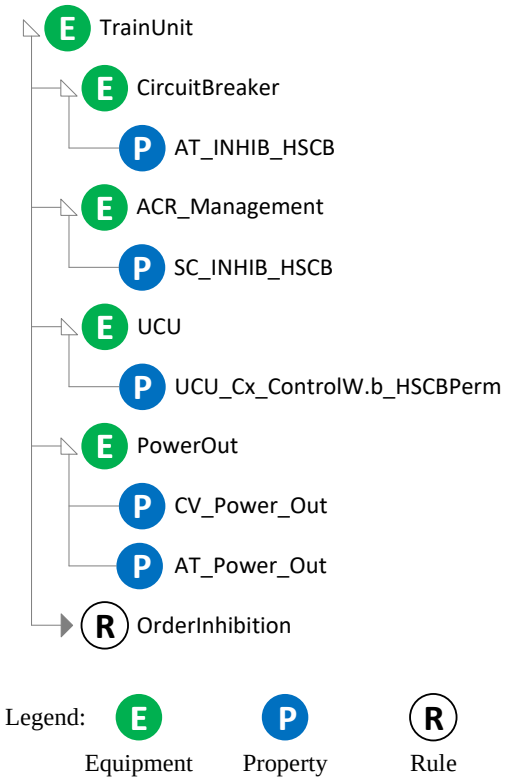

Fig. 2. Software Model

3) Software Model: A software model is the outcome of each exercise when the subject processes the requirements. The resulting software models are corrected by a domain expert, who determines the percentage of steps that are correctly performed within each task. The percentage of correct steps provides the calculation for effectiveness and efficiency. Fig. III-D3 shows a tree representation of the software model and https://www.youtube.com/watch?v=Ypcl2evEQB8 shows the concrete syntax of the software model.

4) Focus Group Interview: The objective of this focus group interview [20] was to obtain qualitative data from comments of the subjects. This focus group interview is composed of open questions. The aim of these questions

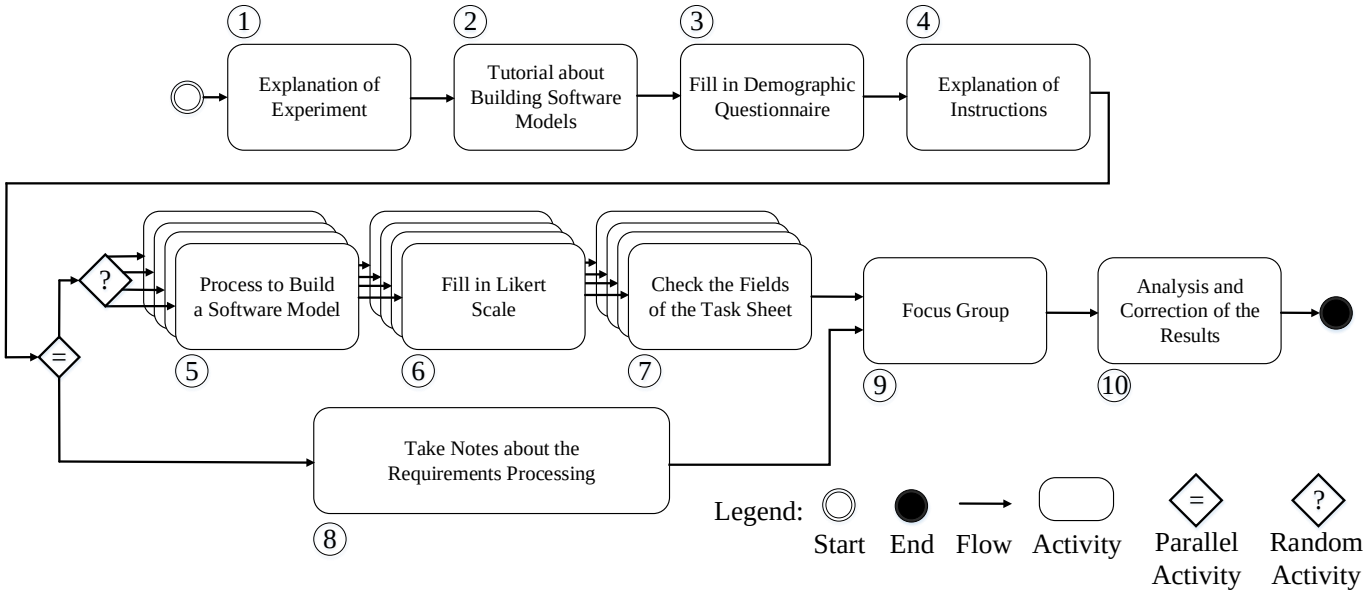

Fig. 3. Procedure of the experiment

is to detect the concepts or processes in the performance of the exercise that are more problematic for subjects as well as to determinate the real causes of the problems. The materials used in this experiment are available at http://svit.usj.es/requerimentinfluenceexperiment.

E. Experimental Procedure

We chose a crossover design where two treatments (enriched and non-enriched requirements) were applied randomly by the subjects. The pros of this design are: (1) it uses the largest possible sample size due to repeated measures; (2) the learning effect is minimized since the treatments alternate; and (3) it is easy to justify in an industrial environment where both treatments are used frequently [22].

In order to verify the experimental parameters, we conducted a pilot study [3] with one participant. This participant did not take part in the experiment later. As a result of the pilot study, we decided to reduce the number of exercises due to the long time that was required to process all of the requirements. Initially, the domain expert designed thirteen requirements for the experiment. After the pilot study, the domain expert reduced the set of exercises to four representative requirements. The criteria for selecting the four requirements was to choose the most important ones for the specification of the software system.

The experiment was conducted on two different days. The first day was performed at the CAF headquarters in Zaragoza (Spain) with a group of six software engineers. The second day, the experiment was replicated at the CAF headquarters in Beasain (Spain) with thirteen software engineers. In this last case, the experiment was performed by three groups of software engineers based their schedule availability. In the first and second groups, there were five software engineers; in the third group there were three software engineers. The procedure (see Figure 3) for all of the subjects was the following:

1) The subjects were given information about the experiment development. They were told that it was not a test of their abilities.
2) The subjects attended a tutorial about how to build the software models according to requirements. This tutorial was taught by the instructor. The average time spent on this tutorial was 24 minutes. The subjects could develop an example during the explanation of this tutorial. Hard copies of the slides that were used in the tutorial were given to the subjects and were available to them during the experiment.
3) The subjects were asked to fill in a demographic questionnaire prior to the experimental tasks.
4) The subjects were then given a series of clear instructions to process the requirements and how to fill out the task sheet.
5) The subjects were asked to interpret four requirements (two enriched requirements and two non-enriched requirements). As a result of this interpretation, the subjects had to build a software model that expressed all the ideas articulated in the requirements. This model was used to calculate the effectiveness and efficiency of the process of interpreting requirements. To avoid a possible ceiling effect, there was no time limit in interpreting requirements. On the other hand, the two types of requirements were assigned to the subjects randomly to avoid the learning effect.
6) The subjects were asked to fill in a Likert scale about perceived difficulty for each requirement. These answers were used to calculate the perceived difficulty about the understandability of requirements.

7) When a subject finished the development of one requirement, before beginning the next one, an observer checked that the subject had filled in all of the fields in the task sheet.
8) The observers took notes about the comments and doubts of the subjects when processing requirements.
9) A focus group interview about the exercises was conducted by the instructor with every group of the subjects.
10) Finally, the domain expert corrected the models and the observers analyzed the results. The percentage of successfully completed exercises provided a value for effectiveness.

IV. Results

For our design, the most suitable statistical test is the Linear Mixed Model [29] test. The Dependent Variables for this test are efficiency, effectiveness, and perceived difficulty. The Fixed Factor is the method, and the Random Factor is the subject since we need to represent the subject of each measure. The conclusions extracted from the Linear Mixed Model are supported with descriptive data through box-and-whiskers plots and histograms.

The use of the Linear Mixed Model test involves the assumption that residuals must be normally distributed. With the collected data, all residuals obtain a p-value that is higher than 0.05 with the K-S test, meaning that residuals are normally distributed. The effect size shows the magnitude of differences for each factor. It is usually applied when null hypotheses are rejected in order to study the level of significant differences between the treatment means. We calculate the effect size through Cohen d, which describes the proportion of the variability in the dependent measure that is attributable to a factor. The most common interpretation is the following: between 0.2 and 0.3 is a small effect; around 0.5 is a medium effect; and more than 0.8 is a large effect [8].

.

A. Effectiveness

A Linear Mixed Model test was conducted to compare the effectiveness with the two different types of requirements. The results show that there is a significant effect of the different requirements on effectiveness [F(2,18) = 559.78 p=0.000]. Since the p-value is less than 0.05, we can conclude that there are significant differences between the two treatments. The effect size of 0.601 shows that the magnitude of this difference is medium.

Fig. 4(a) shows the box-and-whiskers plot for the response variable *effectiveness*. It can be observed that the value for enriched requirements is better than for non-enriched requirements. The median, the first quartile, and the third quartile get better values for enriched requirements.

The average of effectiveness for enriched requirements (86.578%) is also better than for non-enriched requirements (77.605%). Fig. 4(b) shows the histograms for effectiveness with enriched requirements, and Fig. 4(c) shows the histograms for effectiveness with non-enriched requirements.

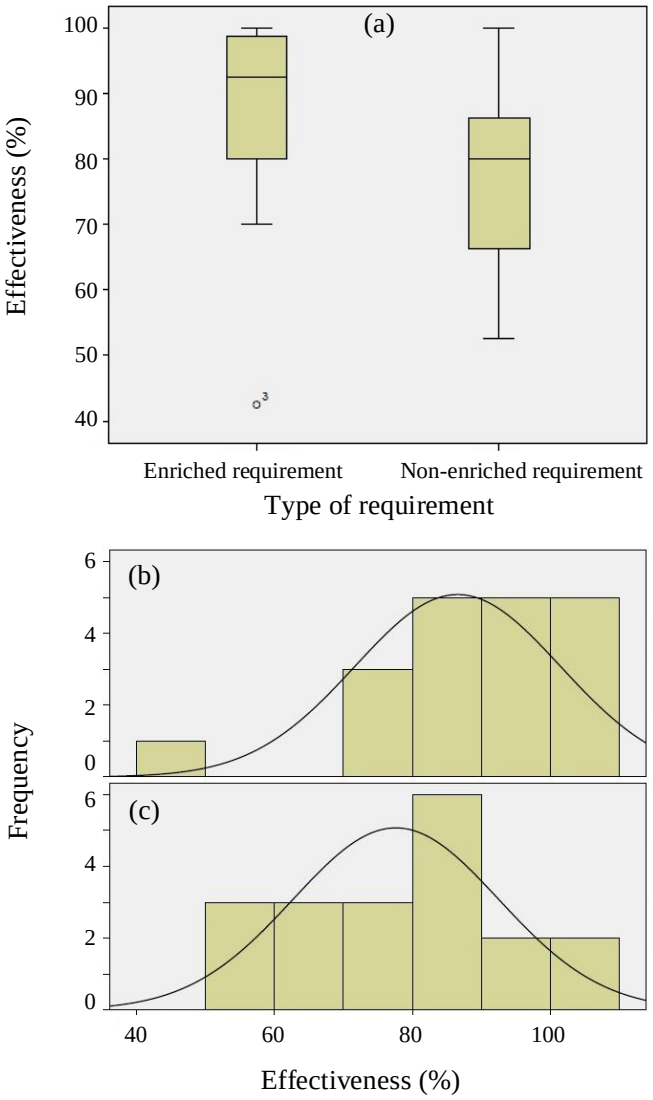

Fig. 4. (a) Box-Plot for effectiveness. (b) Histogram for effectiveness with enriched requirements. (c) Histogram for effectiveness with non-enriched requirements.

Note that an effectiveness of 100% is more frequent in enriched requirements than in non-enriched requirements. There are five samples of 100% in effectiveness when enriched requirements are used, and two samples when using non-enriched requirements. The normal curve shows that values for effectiveness tend to be higher (around 80%) with enriched requirements.

According to our analysis, we can state that there are significant differences between enriched and non-enriched requirements in terms of effectiveness. The values for enriched requirements are better than the values for non-enriched requirements. Therefore, we reject H_{01}, which claims that the effectiveness when working with enriched requirements is the same as when working with non-enriched requirements.

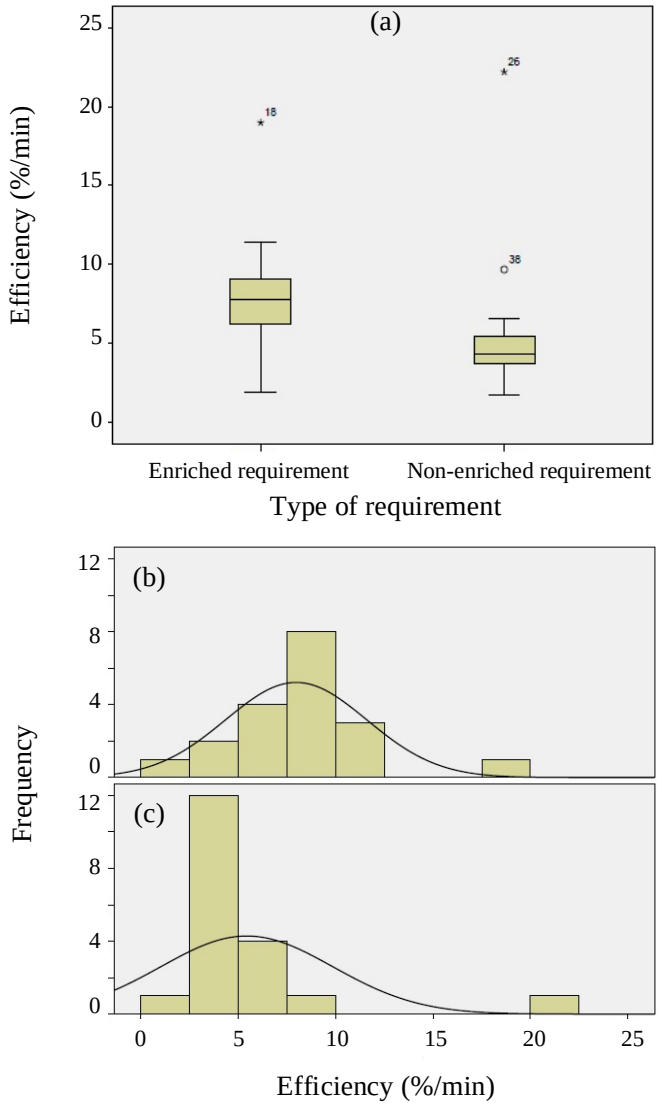

Fig. 5. (a) Box-Plot for efficiency. (b) Histogram for efficiency with enriched requirements. (c) Histogram for efficiency with non-enriched requirements.

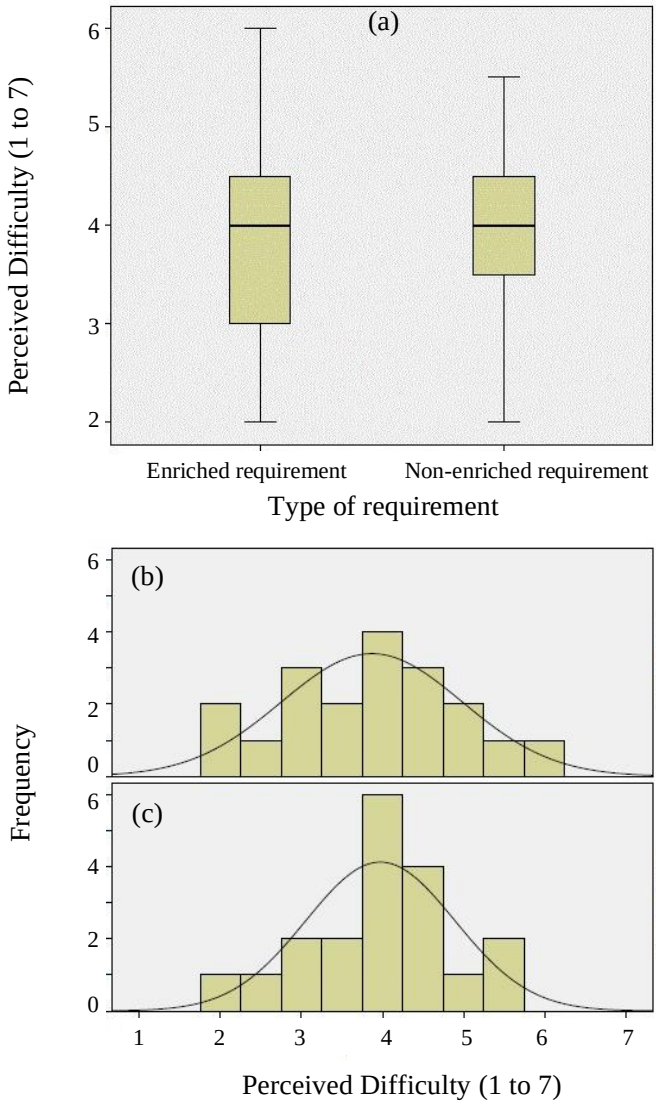

Fig. 6. (a) Box-Plot for perceived difficulty. (b) Histogram for perceived difficulty with enriched requirements. (c) Histogram for perceived difficulty with non-enriched requirements.

B. Efficiency

The results with the Linear Mixed Model test show that there is a significant effect of the two different types of requirements on efficiency [F(2,18) = 56.574 p=0.000] since the p-value is less than 0.05. The effect-size of 0.627 is medium, which means that this difference between the treatments is considerable.

Fig. 5(a) shows the box-and-whiskers plot for the response variable *efficiency*. The median, the first quartile and the third quartile get better values for enriched requirements. Moreover, the average for enriched requirements (7.969%/min) is higher than for non-enriched requirements (5.434%/min). Fig. 5(b) shows the histograms for efficiency with enriched requirements, and Fig. 5(c) shows the histogram for efficiency with non-enriched requirements. It can be observed that most of the samples for non-enriched requirements (12 samples) are between 2.5%/min and 5%/min; on the other hand, the interval with the highest number of samples for enriched requirements (8 samples) is between 7.5%/min and 10%/min. The normal curve shows that values for efficiency tend to be around 8%/min.

According to our analysis, we can state that there are significant differences between enriched and non-enriched requirements in terms of efficiency. The values for enriched requirements are better than the values for non-enriched requirements. Therefore, we reject H_{02}, which claims that the efficiency when working with enriched requirements is the same as when working with non-enriched requirements.

C. Perceived Difficulty

A Linear Mixed Model test was conducted to compare the perceived difficulty of subjects with the two different types of requirements. There is a significant effect of the different

requirements on the perceived difficulty since the p-value is less than 0.05 [F(2,18) = 185.77 p=0.000]. The effect size of 0.102 shows that the magnitude of this difference is small.

Fig. 6(a) shows the box-and-whiskers plot for the response variable *perceived difficulty*. The difference between the averages of the two treatments is small, the average of perceived difficulty for enriched requirements is 3.868, and the average of non-enriched requirements is 3.974. Fig. 6(b) shows the histogram for perceived difficulty with enriched requirements, and Fig. 6(c) shows the histogram for perceived difficulty with non-enriched requirements. Note that the value of means of the two treatments are close. On the other hand, the interval with the highest number of samples (interval around 4) is the same for both treatments. There are four samples in this interval for enriched requirements and six samples for non-enriched requirements.

According to our analysis, we can state that there are significant differences between enriched requirements and non-enriched requirements in terms of perceived difficulty. The values for enriched requirements are better than the values for non-enriched requirements. Therefore, we reject H_{03}, which claims that the perceived difficulty when working with enriched requirements is the same as when working with non-enriched requirements.

V. Discussion

The focus group and the results presented in the previous section enables us to perceive non-enriched and enriched requirements in terms of effectiveness, efficiency, and perceived difficulty as follows:

Effectiveness: Unlike non-enriched requirements, enriched requirements include specific property-value pairs that software engineers can use during the construction of a model from a requirements specification. Nevertheless, the results of our experiment show that properties and values included in an enriched requirement do not always match the properties and values needed in models. In some cases, the software engineers changed the names of the properties, the number of properties used, and the values because they needed to perform intermediate operations to achieve the goal of the requirement.

Even though the software engineers did not include the properties and values in the models as they were in the requirement, we detected that the software engineers used these properties and values as check points, enabling them to check whether the requirement had been fully described. In other words, enriched requirements had *n* properties that the software engineers considered as *n* check points, while non-enriched requirements were considered as a whole. These check points helped the software engineers to fully transfer the requirements to the model.

Efficiency: When asked about their use of property-value pairs to enrich the requirements, the software engineers answered that property-value pairs did not increase the difficulty for customers to read requirements and served CAF engineers as a guide to transfer requirements to models. In other words, (1) natural language provides engineers with flexibility, (2) natural language enables CAF to share the requirements with their customers, and (3) property-value pairs improve the efficiency of software engineers.

Perceived difficulty: Our results showed that the difference in perceived difficulty is small between non-enriched and enriched requirements. The main difficulty in non-enriched requirements is to know their magnitude, whereas the difficulty in enriched requirements is to understand the property-value pairs. The results are aligned with the responses that we obtained during the focus group interview, where the acquired knowledge indicated that the software engineers perceived a similar difficulty between non-enriched and enriched requirements. As a matter of fact, the company did not require from its engineers to use a specific type of requirement nor to have a consensus on which type of requirement was more beneficial than the other. Nevertheless, the effectiveness, efficiency, and perceived difficulty showed a significant difference in favor of enriched requirements, which motivates their usage in the future.

VI. Threats to Validity

This section describes the threats that we have avoided, the threats that we could not avoid but that we mitigated, and the threats that we could not tackle. We use the classification of threats to validity of [27]; this classification distinguishes four aspects of validity:

Construct validity: This type of validity reflects the extent to which the operational measures that are studied represent what the researchers have in mind and what is investigated based on the research questions.

- Author bias: This threats means that the people that define the artifacts can subjectively influence the obtainment of the results that they are looking for. In order to mitigate this threat, the exercises and responses were designed by a domain expert who was external to the design of the experiment and who was not involved in this paper. This expert has developed modeling tools in industrial environments (in the induction hob domain and train control software domain).
- Task design: This threat appears when the tasks can be correctly performed just by chance. To mitigate this threat the proposed exercises did not have a true/false answer; the subjects had to build a model; this is very difficult for subjects to answer correctly if they do not understand the exercise. On the other hand, the requirements statements were real requirements that were extracted from the CAF company's catalog.
- Mono-method bias: This threat is due to using a single type of measure [22]. Satisfaction, effectiveness, and efficiency measurements were affected by this threat. To mitigate this threat for the effectiveness and efficiency measurements, we mechanized these measurements as much as possible by means of exercise decomposition.
- Hypothesis guessing: This threat means that the subject may guess the hypotheses and work to fulfill them. To

mitigate this, we did not talk with the subjects about the research questions or the objective of the experiment.
- Evaluation apprehension: This threat appears when the subjects are afraid of being evaluated. To mitigate this threat the instructor told all of the subjects that it was not a test of their abilities.
- Interaction of different treatments: This threat appears when there are several treatments applied at the same time. To solve this threat, the treatments were applied randomly at different times.
- Mono-operation bias: This threat appears when treatments depend on a single tool only. The experiment was affected by this threat since we worked with a single tool for each exercise. For this reason, the generalization of results must be done with caution.

Internal validity: This type of validity appears when causal relations are examined. There is a risk that the studied response variables may be affected by other factors that are not considered in the experiment.

- Learning effect: This threat appears when the subjects learn something during the experiment that may influence later tasks. We mitigated this threat by randomizing the order of the exercises.
- Information exchange: Since the experiment was designed to take place in two sessions, the subjects might have been able to exchange information during the time between the sessions. This was minimized because the experiment was performed in two different locations on two different days.
- Understandability: This threat appears when the subjects do not understand how to proceed to conduct the experiment. This threat was mitigated by writing the experimental materials in the mother tongue of subjects. In addition, a tutorial about how to build the software models according to requirements was explained by the instructor before the experiment.
- Fatigue effects: This threat appears when the subjects get tired during the experiment. This was solved by establishing a total time of 90 minutes for the whole experiment (including the training). This time was short enough when compared with the regular work time of the subjects.
- Subject motivation: This threat appears when the subjects are not motivated to participate in the experiment. The experiment was affected by this threat since the subjects were recruited as part of their daily work (they were not volunteers).
- Selection: This threat appears when outcomes of the experiment may depend on the type of subjects. The experiment was affected by this threat since all of the subjects were recruited from the CAF company.

External validity: This type of validity is concerned with to what extent it is possible to generalize the findings and to what extent the findings are of relevance for other cases.

- Statistical power: This threat appears when the number of subjects is not enough to generalize results. Our experiment was affected by this threat, because the number of subjects (19) was not high enough to generalize results. However, it is important to note that the role of the subjects (software engineers in an industrial environment) makes an interesting contribution in an area where most experiments are conducted with students. On the other hand, we have tried to use a confidence interval where conclusions are 95% representative. This means that if they followed a normal distribution, the results would be true 95% of the time.
- Influence of the domain: This threat appears when the outcomes depend on a specific domain. This experiment was affected by this threat since we only analyzed the Railway domain.

Reliability: This type of validity is concerned with to what extent the data and the analysis are dependent on the specific researchers.

- Data collection: This threat appears when data collection is not done in the same way throughout the different sessions. This was mitigated by applying the same procedure to each session and using the same formula to calculate the dependent variable values.
- Completion data: This threat appears when there are some missing data after the data collection process. To mitigate this threat, two observers tested the data coherence when the subjects finished each exercise because the subjects themselves wrote down the data used in the metrics of the experiment.

VII. Related Work: Comparison with our Contribution

The process to elicit, report, and use requirements as input for the next steps in the software development process is greatly dependent on human factors. According to Ahmed [1], there are non-technical skills that might affect the process of requirements elicitation. This strong dependency on human skills and on the type of analyst can lead to some requirements specifications being ambiguous in some cases. According to the literature review performed by Bano [4], 81% of papers focus on detecting ambiguity in textual specifications. Bano also states that there is a lack of empirical evaluation of natural language processing techniques for addressing ambiguity in requirements.

Several works have dealt with processing requirements specifications for model building. These works aim to extract conceptual models from texts with natural language requirements. One example of these works was developed by Robeer et al [25], who propose to automatically derive conceptual models from user stories that are written in natural language. The approach is based on an algorithm that combines several heuristics and is evaluated through two case studies. The metrics used in the evaluation are precision and recall to compare the accuracy of the proposed technique with respect to a manual tagging. Bhala and Abirami [28] also proposed an automatic transformation from functional specifications in

natural language to conceptual models. The proposal is based on the analysis of grammatical constructs. The result of the transformation is the construction of an entity-relationship diagram with notations. The proposal is evaluated through a case study that compares the model built through the transformations versus a model built by hand by two users: one expert and one novice. The metric used in the experiment is performance. Ferrari et al. [14] conducted an evaluation of a tool (named CAR) that supports a textual definition of requirements. The evaluation was done using metric completeness, where the experiments compare the completeness of requirements using CAR versus using no tool. The authors of that paper are also the subjects of the study.

There are works that focus on detecting ambiguity in textual specifications, such as Fantechi et al. [13]. These authors defined a set of metrics to perform a quality evaluation of requirements documents that is based on Use Cases templates. These metrics are based on three variables that analyze requirement specifications through linguistic techniques: expressiveness, consistency, and completeness. Linguistic techniques are not sufficient to completely address aspects that are related to the correctness and consistency of requirements. Gnova et al. [16] defined indicators for measuring quality in textual requirements as well as a tool that computes quality measures in a fully automated way. The approach aims to improve the quality of requirements and to improve the writing skills of analysts. Mund et al. [21] conducted two replications of an experiment to study the extent to which software requirements specifications are created and used in practice as well as the degree to which the quality of such specifications affect the next development activities in the process. Quality is measured through a quality questionnaire filled in by professionals in industry. The goal of the experiment is to identify the quality of software requirements specifications.

Apart from the quality of requirement reports, another important issue to consider is how these reports are analyzed in next steps of the software development process. Information reported in documents must be filtered and processed in order to continue with the software development process. Garca-Flores [15] defined semantic filtering techniques for the analysis of large textual requirements descriptions. The approach makes use of a contextual exploration method to identify specific linguistic markers that show the structure of the semantic knowledge. Kof [19] proposed a method to extract ideas from a textual description of requirements using heuristics. The approach consists of a systematic comparison of different term extraction heuristics. According to Kof, heuristics based on named entity recognition are the ones that provide the best performance.

There are several works that have compared two elicitation requirements techniques. One of these works was developed by Besrour et al. [5]. These authors conducted a study to evaluate and compare requirements elicitation techniques using students as subjects. The techniques that are compared are: natural language, a specific requirements template called DIRT, and the IEEE software requirements specification. España et al. [11] compared Use Cases versus Communication Analysis through the framework called Method Evaluation Model (MEM). MEM distinguishes between the actual efficacy and the perceived efficacy of the method. The comparison was conducted using students as subjects and questionnaires. Hadar et al. [18] compared the comprehensibility of requirements models in Use Cases versus Tropos through a family of experiments using students as subjects. The experiments focus on three categories of problem-solving comprehension questions and on the effort required to perform them in terms of the time.

Author	Metrics
Robeer et al. [25]	Precision amnd Recall
Bhala and Abirami [28]	Performance
Ferrari et al. [14]	Completeness
Fantechi et al. [13]	Expressiveness, consistency, and completeness
Génova et al. [16]	Morphological, lexical, analytical, relational
Mund et al. [21]	Quality questionnaire
García-Flores [15]	Semantic metrics
Kof [19]	Heuristics
Besrour et al. [5]	Syntactic, semantic
España et al. [11]	Actual efficacy, perceived efficacy
Hadar et al. [18]	Comprehension an effort

TABLE I
COMPARISON WITH OTHER WORKS

Table I shows a comparison of our study with previous ones conducted in the field of interpreting textual requirements specifications. In general, metrics in existing works focus on measuring quality, syntax, semantics, and time. Moreover, all the empirical studies that have been conducted to compare elicitation techniques have been done with students. To our knowledge, there are no previous papers that have conducted empirical experiments in industry with the aim of comparing different enrichment levels in the requirements specification. Of all of the related works studied, only the experiment of Mund et al. [21] (based on quality questionnaires) was conducted in industry. As Ambreen et al. [2] state, there is a need to conduct empirical studies in industry since the context is not the same as in academia. Thus, our work aims to help cover the dearth of studies that analyze how the description level of the requirements can affect the developer in industry.

VIII. CONCLUSION

Techniques to specify requirements are still a key topic in the software engineering community. Empirical studies need to be conducted in industry since the context is not the same as in academia. In this work, we conducted a crossover experiment to analyze which type of textual requirement description is better for requirements specification: a non-enriched requirement or an enriched requirement. The subjects of the experiment were 19 software engineers from CAF, a worldwide supplier of railway solutions.

The subjects started from an already existing textual description of requirements (non-enriched or enriched) that they had to process to build software models. We measured three variables: effectiveness (ratio of errors), efficiency (time spent), and perceived difficulty (subjective feelings of the software engineers). Our results showed that there are significant differences between enriched and non-enriched requirements for these three variables, demonstrating that effectiveness and efficiency for enriched requirements have better values, while non-enriched requirements are slightly more difficult to deal with than enriched ones. In addition, we detected that the subjects used the property-value pairs of enriched requirements as check points that they should verify in order to determinate whether the requirement was fully processed. More time was required to process enriched requirements than non-enriched requirements, but it influenced effectiveness and efficiency positively.

As future work, we plan to replicate this experiment in more companies in order to generalize the results, independently of the subjects' profile. Moreover, we also plan to analyze enriched requirements for building software models as well as for implementing code.

ACKNOWLEDGMENTS

This work has been partially supported by the Ministry of Economy and Competitiveness (MINECO) through the Spanish National R+D+i Plan and ERDF funds under the project Model-Driven Variability Extraction for Software Product Line Adoption (TIN2015-64397-R). We also thank DataMe (TIN2016-80811-P) from MINECO and IDEO (PROMETEOII/2014/039) from Generalitat Valenciana.

REFERENCES

[1] F. Ahmed. Software requirements engineer: An empirical study about non-technical skills. *JSW*, 7(2):389–397, 2012.

[2] T. Ambreen, N. Ikram, M. Usman, and M. Niazi. Empirical research in requirements engineering: trends and opportunities. *Requirements Engineering*, pages 1–33, 2016.

[3] M. Asadi, S. Soltani, D. Gašević, and M. Hatala. The effects of visualization and interaction techniques on feature model configuration. *Empirical Software Engineering*, pages 1–38, 2014.

[4] M. Bano and N. Ikram. *Addressing the Challenges of Alignment of Requirements and Services: A Vision for User-Centered Method*, pages 83–89. Springer Berlin Heidelberg, Berlin, Heidelberg, 2014.

[5] S. Besrour, L. B. Ab Rahim, and P. Dominic. Exploratory study to assess and evaluate requirement specification techniques using analysis determination requirements framework. *Research Journal of Applied Sciences, Engineering and Technology*, 9(3):165–171, 2015.

[6] J. J. Cappel. Entry-level is job skills: A survey of employers. *Journal of Computer Information Systems*, 42(2):76–82, 2002.

[7] D. Carrizo, O. Dieste, and N. Juristo. Systematizing requirements elicitation technique selection. *Information and Software Technology*, 56(6):644–669, June 2014.

[8] J. Cohen. Statistical power analysis for the behavior science. *Lawrance Eribaum Association*, 1988.

[9] J. Echeverría, F. Pérez, A. Abellanas, J. I. Panach, C. Cetina, and O. Pastor. Evaluating bug-fixing in software product lines: An industrial case study. In *Proceedings of the 10th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*, pages 24:1–24:6, New York, NY, USA, 2016. ACM.

[10] J. Echeverria, F. Pérez, C. Cetina, and O. Pastor. Comprehensibility of variability in model fragments for product configuration. In *CAiSE 2016, Ljubljana, Slovenia, June 13-17, 2016. Proceedings*, pages 476–490, 2016.

[11] S. España, N. Condori-Fernandez, A. González, and Ó. Pastor. An empirical comparative evaluation of requirements engineering methods. *Journal of the Brazilian Computer Society*, 16(1):3–19, 2010.

[12] G. Fanmuy, A. Fraga, and J. Lloréns. Requirements verification in the industry. In *Proceedings of the Second International Conference on Complex Systems Design & Management, CSDM 2011, Paris, 7-9 December 2011*, pages 145–160, 2011.

[13] A. Fantechi, S. Gnesi, G. Lami, and A. Maccari. Application of linguistic techniques for use case analysis. In *Proceedings IEEE Joint International Conference on Requirements Engineering*, pages 157–164, 2002.

[14] A. Ferrari, F. Dell'Orletta, G. Spagnolo, and S. Gnesi. Measuring and improving the completeness of natural language requirements. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 8396 LNCS:23–38, 2014.

[15] J. J. G. Flores. *Semantic Filtering of Textual Requirements Descriptions*, pages 427–433. Springer Berlin Heidelberg, Berlin, Heidelberg, 2004.

[16] G. Génova, J. M. Fuentes, J. Llorens, O. Hurtado, and V. Moreno. A framework to measure and improve the quality of textual requirements. *Requirements Engineering*, 18(1):25–41, 2013.

[17] J. Gonzalez-Huerta, E. Insfran, S. Abrahão, and G. Scanniello. Validating a model-driven software architecture evaluation and improvement method: A family of experiments. *Information and Software Technology*, 57:405–429, 2015.

[18] I. Hadar, I. Reinhartz-Berger, T. Kuflik, A. Perini, F. Ricca, and A. Susi. Comparing the comprehensibility of requirements models expressed in use case and tropos: Results from a family of experiments. *Information and Software Technology*, 55(10):1823 – 1843, 2013.

[19] L. Kof. Requirements analysis: Concept extraction and translation of textual specifications to executable models. In *Proceedings of the 14th International Conference on Applications of Natural Language to Information Systems*, NLDB'09, pages 79–90, Berlin, Heidelberg, 2010. Springer-Verlag.

[20] R. A. Krueger and M. A. Casey. *Focus groups: A practical guide for applied research*. Sage publications, 2014.

[21] J. Mund, D. Mndez Fernndez, H. Femmer, and J. Eckhardt. Does quality of requirements specifications matter? combined results of two empirical studies. volume 2015-November, pages 144–153, 2015. cited By 0.

[22] J. I. Panach, S. España, O. Dieste, O. Pastor, and N. Juristo. In search of evidence for model-driven development claims. *Inf. Soft. Techn.*, 62(C):164–186, June 2015.

[23] S. M. Ratchev, K. S. Pawar, E. Urwin, and D. Mueller. Knowledge-enriched requirement specification for one-of-a-kind complex systems. *Concurrent Engineering: R&A*, 13(3):171–183, 2005.

[24] I. Reinhartz-Berger, K. Figl, and Ø. Haugen. Comprehending feature models expressed in cvl. In *International Conference on Model Driven Engineering Languages and Systems*, pages 501–517. Springer, 2014.

[25] M. Robeer, G. Lucassen, J. M. E. M. v. d. Werf, F. Dalpiaz, and S. Brinkkemper. Automated extraction of conceptual models from user stories via nlp. In *2016 IEEE 24th International Requirements Engineering Conference (RE)*, pages 196–205, Sept 2016.

[26] B. Rosadini, A. Ferrari, G. Gori, A. Fantechi, S. Gnesi, I. Trotta, and S. Bacherini. Using NLP to detect requirements defects: An industrial experience in the railway domain. In *Requirements Engineering: Foundation for Software Quality - 23rd International Working Conference, REFSQ 2017, Essen, Germany, February 27 - March 2, 2017, Proceedings*, pages 344–360, 2017.

[27] P. Runeson and M. Hst. Guidelines for conducting and reporting case study research in software engineering. *Empirical Software Engineering*, 14(2):131–164, 2009.

[28] V. B. R. V. Sagar and S. Abirami. Conceptual modeling of natural language functional requirements. *Journal of Systems and Software*, 88:25 – 41, 2014.

[29] B. T. West, K. B. Welch, and A. T. Galecki. *Linear Mixed Models: A Practical Guide Using Statistical Software, Second Ed.* Chapman and Hall/CRC Press, 2014.

[30] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, and A. Wesslén. *Experimentation in Software Engineering: An Introduction*. Kluwer Academic Publishers, Norwell, MA, USA, 2000.