Specifying Industrial System Requirements using Specification Patterns: A Case Study of Evaluation with Practitioners

Predrag Filipovikj and Cristina Seceleanu
School of Innovation, Design and Engineering, Mälardalen University, Västerås, Sweden

Keywords: Specification Patterns, Pattern-Based Requirements Specification, Industrial Case Study.

Abstract: With the ever-increasing size and complexity of the industrial software systems there is an imperative need for an automated, systematic and exhaustive verification of various software artifacts, such as system specifications, models, code, etc. A potential remedy for this need might lie in a pool of techniques for computer-aided verification of software related artifacts, including system specifications. The Achilles' heel of these techniques, and the main hinder for their wider adoption in the industrial development process are the complexity and the specialized skill-set required for the formal encoding of specifications. To alleviate this problem, *Specification Patterns* that are based on the observation that the system specifications are framed within reoccurring solutions have been proposed. The approach has been shown to be expressive enough for capturing requirements in the automotive domain, however, there is a lack of empirical data that can be used to judge its practical usefulness. In this paper, we involve an existing specification-patterns-based tool, called PROPAS, and propose a small-size evaluation of the approach with practitioners, on a case study conducted in cooperation with Scania, Sweden, one of the world's leading manufacturers of heavy-load vehicles. Our results show that the specification patterns that are supported by an adequate tooling have the potential to be practically usefull for the non-experts in formal methods.

1 INTRODUCTION

Developing and maintaining system requirements specifications (system specifications from now on) for complex software systems is a daunting task. According to the current state-of-the-practice, the system specifications are written using free text, predominantly in English. While the free text natural language specifications are easy to capture and read, such way of specifying systems suffers from a number of limitations. First, the unrestricted natural language specifications lack structure, thus leading to situations where the same requirement can be interpreted in multiple ways, usually depending on the domain knowledge of the person who is reading it. Second, using the unrestricted natural language can potentially lead to an inconsistent encoding. The most prominent example of this kind is when a same concept is referred to by different names.

Such reasons, combined with the ever-increasing safety-critical importance of the functions, call for more precise and unambiguous ways of encoding requirements that will also enable rigorous verification of various models. Existing techniques for computer-aided analysis of software-related artifacts, including the system specifications (Barnat et al., 2016; Post et al., 2011; Filipovikj et al., 2018) are mostly based on formal methods, which have the advantage of being completely automated, mathematically rigorous and possibly exhaustive. Despite their potential added value, the formal analysis techniques are still just seldomly applied in industrial practice. One of the main obstacles towards a broader industrial adoption of such techniques seems to be the formal encoding of the textual specifications in some form of temporal logic, which is a challenging task for practitioners, without appropriate tool support.

Specification Patterns (SPS) (Dwyer et al., 1998) is an approach intended to help practitioners to formally encode system specifications. The approach is based on reoccurring solutions for specifying requirements, which have been collected in a catalogue of patterns. The initial set of patterns has been extracted by analyzing more than 500 requirements from various domains. The advantage of the specification patterns is that they have a well defined structure and semantics, and as such can be encoded in many differ-

*corresponding author

Filipovikj, P. and Seceleanu, C.
Specifying Industrial System Requirements using Specification Patterns: A Case Study of Evaluation with Practitioners.
DOI: 10.5220/0007726600920103
In *Proceedings of the 14th International Conference on Evaluation of Novel Approaches to Software Engineering (ENASE 2019)*, pages 92-103
ISBN: 978-989-758-375-9

ent ways, including the constrained natural language and various formal notations that accurately capture the semantics of the system behavior described by the particular requirement.

Besides the body of work focusing on enriching the specification pattern catalogue (Dwyer et al., 1998; Konrad and Cheng, 2005; Grunske, 2008; Autili et al., 2015) and providing specialized tool support (Smith et al., 2002; Mondragon and Gates, 2004; Remenska et al., 2014), there exist several endeavors targeting the expressiveness of the specification patterns in the automotive domain, showing that the specification patterns are expressive enough for formalizing the requirements in the domain (Post et al., 2012; Filipovikj et al., 2014). However, in all studies the generation of the formal system specifications based on the specification patterns is performed by formal methods experts. Consequently, there is a lack of empirical data when it comes to applying the approach in industrial settings, by the intended users, that is, the engineers.

In this paper, we report the results from an exploratory case study, which is intended to collect data about the practical usefulness of the specification patterns in industry. The exploratory nature of the case study means that the generated results are to be used for proposing hypotheses that can then be tested in future studies. The case study is performed in cooperation with Scania, one of the world's leading manufacturers of heavy-load vehicles (trucks and buses), as part of an ongoing academic-industrial cooperation. In this case study, we involve five different stakeholders of the system development process, as case-study subjects (CSS), who use the specification patterns and the existing tool called PROPAS (Filipovikj et al., 2018) to specify a predefined set of requirements of an operational automotive system. For the purpose of our study, we collected both quantitative and qualitative data. The quantitative data consists of: i) *specification correctness*, which is measured in terms of how many of the requirements are correctly specified using the specification patterns and the PROPAS tool, and ii) the *required time for specification*. The qualitative data on the other hand, is collected via a survey of ten statements, which can be answered using five predefined options.

The paper continues as follows. In Section 2 we present an overview of the specification patterns (Section 2.1) and the PROPAS tool (Section 2.2). Next, in Section 3 we provide details on the case study research method, followed by Section 4 in which we show the details about the setup of our case study. In Section 5 we present the results of the evaluation and the threats to validity, followed by the discussion of the results in Section 6. We compare to related work in Section 7, and in Section 8 we outline the conclusions and directions for future work.

2 SPECIFICATION PATTERNS AND PROPAS TOOL

In this section, we give an overview of the specification patterns (Section 2.1) and the PROPAS tool (Section 2.2).

2.1 Specification Patterns

The *specification Patterns System* (Dwyer et al., 1998) is an approach proposed to facilitate the formal specification of system properties for practitioners who are not experts in formal methods. It is based on the assumption that the systems' specifications are framed within reoccurring solutions, from which a set of patterns can be extracted and saved for future reuse. Each pattern is characterized by a *behavior* that it captures, and an extent of the program execution, called *scope*, within which the behavior must hold. The patterns are expressed as a combination of literal and non-literal terminals. The non-literal terminals can be either Boolean expressions that describe system properties, or integer values that capture timing aspects. The rest of the pattern is made of literal terminals, which represent the fixed part of the pattern and cannot be changed.

The original SPS catalog proposed by Dwyer et al. (Dwyer et al., 1998) is compiled by analyzing more than 500 examples of property specifications from various domains. It contains 13 qualitative patterns, which for easier navigation are divided into two categories: *order* and *occurrence*, expressed in various types of temporal logic, including but not limited to (Timed) Computation Tree Logic ((T)CTL) (Alur et al., 1993), Real-time Interval Logic (RTGIL) (Moser et al., 1997), Metrics Temporal Logics (MTL) (Alur and Henzinger, 1993), etc. The patterns in the later category describe the occurrence of a given event in the system, while the patterns from the ordering category are used to capture the relative ordering of the occurrence of multiple events during system execution. In order to be able to specify real-time systems properties, Konrad and Chen (Konrad and Cheng, 2005) have subsequently extended the specification patterns catalog with a new category of patterns called *real-time*. Consequently, the new and extended catalog was named Real-time Specification Pattern System (RTSPS). Besides the introduction of the new set of patterns, in the same work, Konrad and

Cheng proposed an additional encoding of the specification patterns in controlled natural language (CNL). The intended purpose of the CNL representation is to enable various stakeholders who are not trained in formal methods to read and interpret the system specification expressed via patterns. For instance, a bounded-response pattern over events P and S with global scope expressed in CNL, reads as follows:

Globally, it is always the case P is followed by S within T seconds.

Based on the formal definitions for both the scope (*Globally*) and the behavior (*bounded-response*) (Dwyer et al., 1998; Konrad and Cheng, 2005), one can automatically generate the following TCTL property specified using the CNL pattern above:

$$AG(P \Rightarrow AF_{\leq T}S)$$

The introduction of the real-time patterns did not yield the catalog of patterns complete. In order to satisfy the different purposes, the catalog has since been extended in many directions. The most notable ones include the introduction of *probabilistic patterns* by Grunske (Grunske, 2008), which provide means for one to specify probabilistic system properties. The most comprehensive catalog compiled to date is by Autili et al. (Autili et al., 2015). In our work, we use the following subset of real-time specification patterns:

P1: Globally, Universally: *Globally, it is always the case that P holds.*

P2: Timed Globally, Universally: *Globally, it is always the case that if P held for T time units, then S holds.*

P3: Globally, Real-time Response: *Globally, it is always the case that if P holds, the S eventually holds within T time units.*

P4: After Q Until R Universally P: *After Q, it is always the case case that P holds until R holds.*

P5: Timed After Q Timed Until R Universally P: *After Q holds for T time units, it is always the case that P holds until R holds or at most T_1 time units.*

2.2 ProPaS Tool

The PROPAS tool (Filipovikj et al., 2018) is an open-source tool that is used for the formal system specification using specification patterns. In the following, we overview in details the user interface (UI), as well as other non-UI-related features that have been introduced to support our experiment.

The UI of the PROPAS tool is given in Figure 1, and it represents an improvement of an earlier tool called SeSAMM Specifier (Filipovikj et al., 2016). In the top left part of the UI we have the list of all the

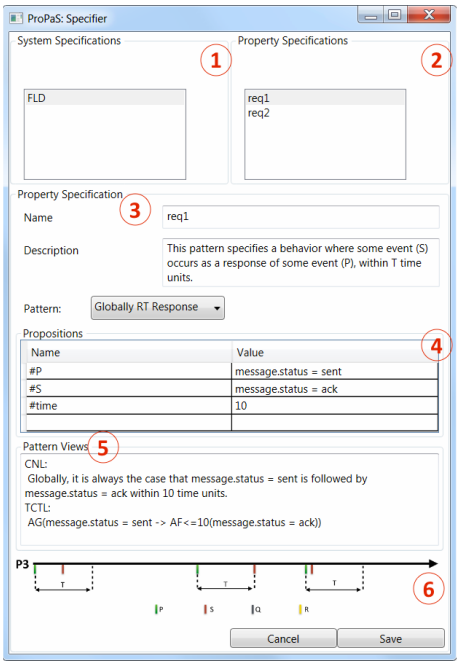

Figure 1: ProPaS Tool User Interface.

system specifications (denoted with 1). The list can be manipulated via commands for deleting an existing specification or creating a new one, which are provided by a context menu that is available on right click of the list. Next to it, the list of properties in the currently active system specification are displayed. Similarly as for the list of system specifications, one can manipulate the list of properties (2) using the context menu that is displayed upon right-clicking the list. The currently available actions over the list of properties include removing an existing property or adding a new one.

The middle part of the UI (3) contains details about the currently active system property. The details are composed of editable and read-only attributes. The first editable attribute for the system property is its *name*, which is also used as an identifier. In the particular example illustrated in Figure 1, the name of the currently active property is *"req1"*. Next comes the first read-only attribute, called *description*, which is defined by the selected pattern. The *specification pattern* for the property is applied from the drop-down list, which contains all the available patterns that the tool has access to. Below the list of available specification patterns, a list of *propositions* (4) characteristic for the specific pattern is given. In the bottom part of the UI (5) there are two elements that serve the purpose of giving feedback to the user on the specified behavior. First, there is a text field in which the combination of the property and the pattern specific attributes are combined to automatically

generate the different views for the given property. In this case, the specification pattern is encoded in constrained natural language and TCTL. Lastly, the UI shows a visual representation (6) of the ordering of the events as specified by the selected pattern, which in the example given in Figure 1 is the ordering of the events according to the bounded-response pattern.

Beside the UI, the PROPAS tool provides other features, out of which some are relevant and some are not for the case study. Out of the study-relevant features, the most important one is the *timer* feature, which is used to keep track of the amount of time that is spent on specifying each property of the set. Out of the non-relevant features worth mentioning is the consistency analysis feature, which has been proposed and implemented in the previous version of the tool. For more details on the consistency analysis, we refer the reader to earlier work (Filipovikj et al., 2018).

3 RESEARCH METHOD

In this section, we present the case-study research method, which is used in the evaluation of the specification patterns and the PROPAS tool.

A *case study* is a research method in software engineering research that aims at studying a contemporary phenomena in their natural context (Wohlin et al., 2012). Initially the term case study was used to denote a demonstration case, during which an implementation of some tool or other type of concept is presented. Subsequently, it got a broader meaning and is also used to refer to self-experienced or self-reported investigation.

The case-study research method is mostly used to provide deeper understanding of the phenomena under study. According to Runeson and Höst (Runeson and Höst, 2009), a case-study research method is *"well suited for many kinds of software engineering research, as the objects of study are contemporary phenomena, which are hard to study in isolation"*. In general, the case-study research method represents a more liberal way of studying some phenomena related to a single project. It is easy to plan and execute, and as such, it is useful for evaluating new methodologies and tools in industrial settings. On the other side, compared to other empirical research methods, such as for instance the *controlled experiment* (Wohlin et al., 2012), the results obtained from a case study are generally more difficult to interpret and generalize. Despite the limitations, if a well-established case-study research methodology is followed, then the obtained results are valuable.

Relevant literature (Runeson and Höst, 2009) suggests that any well-designed case study should follow at least the following five process steps: i) case study panning and design, ii) data collection preparation, iii) data collection, iv) data analysis v) reporting. In order to better plan and execute the listed steps of a case study, there are several papers that provide comprehensive check lists (Kitchenham et al., 1995; Runeson and Höst, 2009), which contain series of questions intended to guide the researchers to correctly plan and execute the research process.

During the planning and design phase, the most important thing is to correctly determine the objective of the study. In software engineering, in general there are four predominant types of case-study research: *exploratory*, *descriptive*, *explanatory* and *improving* (Robson and McCartan, 2016). Basically, the type of the case-study research determines the characteristics of the subsequent steps. For instance, if there is a lack of empirical data on the subject, one resorts to an exploratory case study where the main objective is to provide information on what is happening, and generate ideas and hypothesis for new research. Another important aspect to consider during the planning and design phase is to account for the trade-off between the level of control over the experiment and the degree of realism. Given that the primary focus of a case-study research process is to study some phenomena in their natural context, it is expected to lose some control in favour of a high degree of realism.

In a case-study research, there are two types of variables: *independent* and *dependent*. Independent variables are those variables that can be controlled and changed during the experiment. They must have some effect on the dependent variables. The dependent variable(s) a.k.a. *response variables* are the variables that cannot be controlled in the experiment, and they are usually measured based on the independent variables.

The data collected in a case study is either *qualitative* or *quantitative*. Quantitative data is represented with numbers, whereas the qualitative data is usually expressed in words, descriptions, diagrams, etc. Due to its descriptive nature, the qualitative data is usually preferred in case-study research, however, a combination of qualitative and quantitative data provides better understanding of the phenomena under study. There are three well-established methods for data collection in case study, namely: direct, indirect and independent (Runeson and Höst, 2009). Regardless of which method is collected for the study, it is recommended that multiple data sources are considered in order to reduce the effects of the interpretation bias.

When it comes to the data analysis, different tech-

Table 1: Case study subjects and their roles.

CSS	Role
CSS_1	Software architect for autonomous systems
CSS_2	Development engineer (former quality assurance engineer)
CSS_3	Development engineer/researcher
CSS_4	Requirements Coach
CSS_5	Information/safety architect

niques apply for qualitative and quantitative data. The accepted analysis methods for quantitative data include: descriptive statistics, correlation analysis, development of predictive models and hypothesis testing, whereas the qualitative data are usually analyzed using categorization or sorting.

Finally, the report of the case study serves the purpose of communicating the case-study results. It is often the case that the report cannot be distinguished from the case study itself, and as such it is the main source for judging the overall quality of the study. For communicating results from a case study to a research community the preferred forms of reporting include conference or journal articles or in some cases even technical reports.

4 CASE STUDY PLANNING AND EXECUTION

In this section, we present the details about the planning and execution of our case study, in which we evaluate the PROPAS tool and the specification patterns based on their application by the industrial practitioners.

4.1 Case Study Design

There is a lack of empirical data on whether the specification patterns can be effectively and efficiently used by the engineers and other stakeholders for specifying their systems. Given this, the objective of our case study is to be exploratory, that is, to provide an initial set of empirical data that can be used later to form hypotheses and execute larger case studies in various companies and domains. For planning and executing the case study, we are mainly guided by the checklists for case-study planning and execution by Kitchenham et al. (Kitchenham et al., 1995) and Runeson and Höst (Runeson and Höst, 2009).

For the case study, we consider an operational system, namely the Fuel Level Display (FLD), which is installed in all heavy load-vehicles produced by Scania, Sweden. The core of the system is a software module, which is responsible for: i) reading the value of the sensor installed inside the fuel tank, ii) calculating the estimated total available fuel left in the tank, and iii) showing the estimated value on the dashboard to the driver. The specification of the FLD system consists of twenty four requirements. As the intended functionality of the system is not of a primary interest for this case study, we omit further details. For a detailed system description, we refer the reader to the paper by Westman and Nyberg (Westman and Nyberg, 2014).

The main objective of this study is to collect data that can be used to form a future hypothesis regarding the practical usefulness of the specification patterns and the PROPAS tool for specifying requirements in the automotive domain. The data gathering phase of the case study has been executed at the company premises, involving five engineering professionals from Scania as case-study subjects, each having a different role in the systems development process. The roles for each of the subjects are given in Table 1. For the purpose our our study, the subjects have been asked to specify the following subset of FLD requirements using a predefined set of specification patterns (see Section 2.1) and the PROPAS tool:

R_1 *If actualParkingBrake is false, then the indicatedFuelVolume showed by the fuel gauge is less or equal to actualFuelVolume.*

R_2 *If CAN1 is equal to a CAN message rmsg, then rmsg is in HWreceiveBuffer and Rcan_decodeCan is set to true.*

R_3 *The position of the floater sensedFuelLevel, sensed by the fuel sensor, does not deviate more than 10% from actualFuelVolume in the fuel tank.*

R_4 *If CAN1 is equal to a CAN message rmsg, then rmsg is in FiFoBuffer within 20ms.*

R_5 *If the DMA channels timerCh and rfifoCh are enabled for approximately 20ms, then a RAW value of VFuel is available in ADC_RFIFO.*

R_6 *If actualParkingBrake is false and CAN1 is equal to CAN message FuelEconomy and it has not passed more than 300ms since the last time CAN1 was equal to FuelEconomy, then the CAN signal FuelRate in FuelEconomy does not deviate more than 1% from the derivative of actualFuelVolume.*

The given requirements represent a diverse subset from the FLD functional requirements. The requirements capture both the untimed (R_1, R_2 and R_3) and timed (R_4, R_5, R_6) system behavior.

The case study is executed in three major phases. In the first phase, we compiled a list of potential case-study subjects and contacted them personally in order to obtain their consent for participation in the study.

Upon our request all of the potential case-study subjects agreed to participate. Since none of the subjects has previous knowledge or experience in using the specification patterns and the PROPAS tool, we prepared and distributed to each of them a short tutorial. The tutorial, in a form of a written document, contains a compact description about the general idea of specification patterns, the set of requirements intended for the case study and a set of step-by-step instructions (complemented by screen shots of each step) on how to use the PROPAS tool. During the second phase of the case study, which is the data collection phase, one of the researchers met with each of the subject in person in order to carry out the given task of specifying the predefined set of requirements using the PROPAS tool. The last phase of the case study included data analysis, drawing conclusions and producing the case-study report.

As independent variables in our case study we identify the following: i) the system (FLD) and the subset of the functional requirements used for specification, ii) the PROPAS tool and the set of patterns, and iii) the case-study subjects. The set of dependent variables consists of the following: i) correctness of the specification, ii) required time for specification, and iii) level of domain expertise of the case-study subjects.

4.2 Data Collection Preparation

The quantitative and qualitative data in this case study is collected from more than one source. For collecting the qualitative data, we use a direct method facilitated by the *"save to disk"* feature of the PROPAS tool, which enables the subjects to save their system specifications on a computer hard drive. The data produced by the PROPAS tool includes a pattern-based specification of the predefined set of requirements augmented with information on how much time was spent for specifying each individual requirement.

For qualitative data, we use a survey composed of the following ten statements:

S1 *I can specify the given set of requirements using the specification patterns (SPS).*

S2 *The SPS formalized patterns expressed in constrained natural language are comprehensible.*

S3 *The visual representation of the pattern is helpful in understanding the behavior represented by the selected pattern.*

S4 *The* PROPAS *tool is intuitive and easy to use.*

S5 *The* PROPAS *tool is practically useful for specifying requirements using SPS.*

S6 *The quality of generated specifications is high.*

S7 *I feel confident using the* PROPAS *tool.*

S8 *I am able to use the tool without any additional support.*

S9 *The tool is helping in decreasing the specification effort.*

S10 *I can easily select the appropriate pattern for a given requirement.*

Each of the above statements is intended to capture the subjects' opinion in terms of degree of satisfaction, via the following set of predefined options: i) *"I strongly agree"*, ii) *"I agree"*, iii) *"I neither agree or disagree"*, iv) *"I disagree"* and v) *"I strongly disagree"*. For collecting the qualitative data, we use the online Google Survey. Some of the main advantages of using this tool for collecting the qualitative data are: fast and straightforward survey generation, distributed access to the questions, real-time access to the data once entered by the case-study subjects and the convenient visual representations of the results offered by the tool.

4.3 Data Collection

The data collection was performed according to the data collection plan described in Section 4.2. The qualitative data was collected in real-time during the study execution through the PROPAS tool, at Scania premises. To avoid technical problems, such as software compatibility that includes the operating system and the run-time environment required by the PROPAS tool, the tool was pre-installed on a laptop provided by the researchers. Then, the machine with the PROPAS tool was used by each of the case-study subjects to perform the system specification and to fill in the survey. During the data collection phase, only the current case-study subject and one of the researchers involved in the case study were present in the room. The subjects were expected to use the tool on their own, based on the information from the previously provided tutorial. However, during the specification process, the subjects were allowed to interact with the researcher/expert in order to improve the accuracy of their results and increase the confidence in their work. Each interaction initiated by the case-study subject was recorded by the researcher, as part of the qualitative data that characterizes the practical usefulness of the method and the tool. In general, the less the interaction between the subject and the expert, the higher the understanding of the subject regarding the method and the tool.

After the subjects declared that they were done with the specification of the predefined set of require-

ments, they were asked to fill in the survey with qualitative data. In addition to the survey, we collected the reflections and the comments of the subjects expressed while performing the system specification task using PROPAS.

After both the system specification and the survey were completed, the data collection phase was concluded for that particular subject. At the end of the data collection phase, the following data was collected: five system requirements specifications containing the predefined set of requirements from the FLD system complemented with the amount of time spent per requirement, and five survey responses to our survey for the qualitative data.

4.4 Data Interpretation and Analysis

The third, and the last phase of the case study is the data interpretation and analysis. The gathered quantitative data is analyzed using statistical methods. In order to determine the correctness of the specification, that is, to determine whether a correct specification pattern is used for a given requirement, we use a baseline system specification of the same set of requirements, which was created by a formal methods expert.

The gathered qualitative data on the other hand, is interpreted in a less formal way, mainly through a discussion of the obtained survey data. Given that the primary purpose of the qualitative data is to provide us with a subjective reflection of the case-study subjects on the whole process, the discussion is mainly intended to draw conclusions and extract suggestions that can be useful for setting up future similar case studies, but also for the researchers in order to further improve the PROPAS tool or any other tool designed for the purpose of specifying (formal) requirements.

5 RESULTS

In this section we show the results obtained from the case study. In Section 5.1 we present the quantitative data, followed by the qualitative data results in Section 5.2, after which we discuss the potential threats to validity of our results in Section 5.3.

5.1 Quantitative Data Analysis

The collected quantitative data in the case study, represented through the correctness of the specifications produced using the specification patterns and the time for specification are given in Tables 2 and 3, respectively.

Table 2: Pattern-based requirements specification correctness.

CSS	R_1	R_2	R_3	R_4	R_5	R_6	**total**
CSS_1	✓	✓	✓	✓	✓	✓	100%
CSS_2	×	×	✓	✓	✓	✓	66.67%
CSS_3	✓	✓	✓	✓	✓	✓	100%
CSS_4	✓	✓	✓	✓	✓	✓	100%
CSS_5	✓	✓	✓	✓	✓	×	83.33%
total	80%	80%	100%	100%	100%	80%	90%

Table 3: Requirements specification time in minutes.

CSS	R_1	R_2	R_3	R_4	R_5	R_6	**total**
CSS_1	3	2	2	2	4	13	26
CSS_2	3	6	2	2	1	16	30
CSS_3	2	2	2	2	2	16	26
CSS_4	3	2	2	2	2	18	29
CSS_5	2	3	2	2	2	10	21
avg.	2.16	2.50	1.66	1.66	1.83	12.16	26.4

The correctness of the produced system specifications given in Table 2, shows the data both per case-study subject and the individual requirements, respectively. In the same table, we denote by the symbol (✓) that a given subject has specified a particular requirement correctly, meaning that the appropriate patterns for specification has been selected, and the result is equivalent with the baseline specification. The opposite case, that is, when an incorrect pattern is used is denoted by the symbol (×). By analyzing the specification correctness per subject, we observe the following: three of the subjects achieve 100% correctness, that is, for each of the six requirements a correct specification pattern is used. One of the subjects selected an incorrect pattern for requirements R_1 and R_2, thus achieving (66.67%) correctness of the specification. Lastly, one subject used an incorrect pattern for one requirement, namely R_6, resulting in 83.33% correctness of the produced specification.

If we analyze the collected data per requirement, we observe a single mistake (80% correctness) in the specification of the requirements R_1, R_2 and R_6. The rest of the requirements are specified correctly (100% correctness) by all of the case-study subjects. If we consider the total number of instances of the specified requirements, which is thirty and compare them to only three mistakes, the overall conclusion is that we have 90% correctness for the specification of all requirements by all subjects.

The second aspect of the collected quantitative data is the specification time, which is given in Table 3. The table shows the specification time in minutes. Similarly as for the correctness of the specification, the results for measured specification time can be analyzed both per subject and per requirement. Analyzing the specification time per subject, shows that

the shortest time for specifying the complete set of requirements is 21 minutes, whereas the slowest one is 30. The average specification time for the set of the requirements per subject is 26.4 minutes. The average time per requirement is 4.4 minutes, or more accurately around 2 minutes for R_1 to R_5, whereas for R_6 is around 12 minutes, which accounts for almost half of the time spent on the entire specification.

5.2 Qualitative Data Analysis

The qualitative data in the case study is collected in two ways: through a survey and by taking notes of comments by the case-study subjects during the data collection process. While the data collected from the survey is based on answers to a predefined set of statements, the notes taken during the data collection are based on comments by the subjects and as such are intended to replace the free-text question in the survey. The intended purpose of the qualitative data collection in this case study is to give us a more clear picture on how the subjects perceive the system specification using the specification patterns and the PROPAS tool, and to point to some directions for future work with respect to both the method and the tool, such that their practical usefulness is improved.

Based on the responses of the subjects, we have the following quantitative data:

- **S1** 20% strongly agree, 60% agree, 20% neither agree or disagree
- **S2** 60% strongly agree, 40% agree
- **S3** 20% strongly agree, 20% agree, 60% neither agree or disagree
- **S4** 20% strongly agree, 80% agree
- **S5** 40% strongly agree, 40% agree, 20% neither agree or disagree
- **S6** 60% strongly agree, 40% neither agree or disagree
- **S7** 20% strongly agree, 40% agree, 20% neither agree or disagree, 20% disagree
- **S8** 40% strongly agree, 40% agree, 20% neither agree or disagree
- **S9** 40% strongly agree, 60% agree
- **S10** 20% strongly agree, 20% agree, 60% neither agree or disagree

The obtained qualitative data can be interpreted in various ways, however, it is clear that the case-study subjects are positively inclined towards the specification patterns and the PROPAS tool, which can be concluded based on the fact that majority of the statements are described using positive to neutral phrases. The only negative description can be observed for the statement S_7, where 20% of the subjects express that they do not feel confident in using the PROPAS tool.

The qualitative data obtained through the survey is additionally complemented with unrestricted feedback provided by the case study subjects while they are performing the specification using the PROPAS tool. Since the case-study subjects were not explicitly asked to provide this kind of feedback, CSS_3 and CSS_5 completed the specification task without giving any feedback. The rest of the subjects provided their opinion either during the specification process or immediately after the specification phase was concluded. The provided feedback is not extensive, but it is very important, as the case-study subjects were free to elaborate on the aspects of both the method and the tool that were not reflected by the survey. From the obtained feedback, we point out the following: CSS_1 was positive during the specification of the requirements. The subject pointed out that the method is useful, however, the subject also added some suggestions for improving the tool. Concretely, it has been pointed out that in order for a tool such as PROPAS to be considered for adoption in in the industrial development process of automotive systems, it should also provide requirements management features such as versioning, version comparison, etc. The reflection of CSS_2 is very interesting as the subject has phrased his experience with the specification patterns and the PROPAS tool as follows: *"I can write requirements in natural language as free-text much faster, but the pattern-based requirements specified via the* PROPAS *tool are much more concise"*. Lastly, similar to CSS_1 and CSS_2, CSS_4 also had a positive incline towards the method and the tool, however, the subject pointed out that this version of the tool must be improved in order to be considered for industrial adoption. Especially, it has been pointed out that the visual feedback mechanism should be coupled with the specified requirement and not with the pattern as it is now in the tool. Similar comment has been given by CSS_1.

5.3 Threats to Validity

In this section we discuss the threats to validity of the results obtained in this case study. First, we discuss the internal threats to validity and then the external ones.

Internal Threats to Validity are often defined as *influences that can affect the independent variables with respect to causality without the knowledge of the researchers.* Based on this definition, we identify the following internal threats to validity for our case study: i) case-study subjects selection, and ii) the number of requirements used in the case study.

The selection of the case study subjects is done in such a way as to satisfy a number of criteria, including the following: i) have at least one representative of each category of stakeholders involved in the system development process, and ii) have subjects with various levels of experience and expertise. The case study subjects and their roles are given in Table 1.

Given that there are number of Scania employees satisfy the above criteria, we additionally took into consideration the following factors, which are not relevant for the validity of the results: the availability of the subjects at a specific time and their willingness to participate in the case study.

Due to the limited time that the case-study subjects were assigned for the participation in the case study, we were required to isolate a small subset of all the possible requirements. For that purpose, we compiled a set of requirements that are feasible to be specified within the given time frame of around one hour. Regarding the selection of the system and the requirements to be used in the case study, we had to take into consideration several factors such as: complexity of the system and its specification, and the possibility to make the requirements public.

External Threats to Validity represent the conditions that limit the ability to generalize the results from the case study over the whole domain. In our case study, we consider the following two factors as external threats to validity: i) the selection of the particular system and the particular set of requirements used in the study, and ii) the relative domain expertise of the case study subject compared to their peers from the same company or from the companies within the automotive domain.

The system used in the case study and the rather small set of requirements hinder us from the generalizing the obtained results, however, the obtained results serve well the purpose of an exploratory case study aimed at gathering empirical data. There are several concerns that arise from the system and the mentioned selections, such as: will the results be similar if a different system would be used? Will results be the same if more requirements of the same system were used or more requirements but from different system(s)? In order to answer these questions, a larger case study that includes more systems and case study subjects is required.

The second external threat to the validity of the results is the relative expertise of the case study subjects compared to their peers, be they from the same company or from a different one. We have no means nor a set of well-defined parameters on how to measure the expertise of the case-study subjects. Consequently, in order to be able to pose and test hypotheses for the whole domain, one needs to include more case study subjects preferably from different companies in the same domain, or at least to find an objective metrics to quantify the expertise of the case study subjects.

6 DISCUSSION

In this section, we discuss both the quantitative and the qualitative data obtained from the case study. Based on the discussion, we draw some conclusions which can be useful in defining future hypothesis.

The collected quantitative data shows that on average the industrial practitioners achieve 90% correctness of the specification, while spending 4.4 minutes on average per requirement. However, a more in-depth analysis of the results shows some surprising trends. The selected group of requirements are not of equal complexity. For instance, the simplest requirements specify instant causality (such as in R_3), whereas the most complex one (R_6) specifies complex behavior captured by occurrence of two events. Intuitively, one would expect that most of the mistakes in the specification to occur in specifying the most complex requirements, nevertheless, our data does not provide evidence to support this expectation. If we refer back to Table 2, we can see that three mistakes made in the specification have been connected to different requirements, with only one being for R_6. When it comes to required time for specification, the data given in Table 3 shows that the time required for specifying the requirements is proportional to their complexity. The average time required for specifying R_6 is 12.1 minutes, which is almost half of the total average time for the whole specification.

Some of the subjects employed interesting strategies/heuristics when it comes to selecting the appropriate pattern and identifying the propositions for a given requirement. For instance, CSS_2 selected the wrong pattern, that is, P3 instead of P1 for both R_1 and R_2. Given the fact that P3 by definition incorporates three propositions, compared to only one in P1, the subject introduced new data that does not exist in the original requirement (the maximum response time) in order to fill in the missing propositions. CSS_4 also employed an interesting heuristic for selecting a pattern for a specific requirement, as follows: first, when reading the requirement, the subject tried to determine whether the requirement incorporates timing information or not. Given that in the case study only one of the pattern is untimed (does not encode time explicitly), for requirements that do not incorporate time the subject selected P1 without much hesitation. For the rest of the requirements, which are timed, the subject

analyzed the structure of the pattern by propositions that model events and time and tried to match it with the information encoded in the given requirement.

The analysis of the qualitative data shows that the industrial practitioners are generally favourable for the specification patterns. This is supported by the fact that they have expressed positive opinions about the method and the quality of the generated system specifications. However, based on the qualitative feedback, some issues, primarily related to the tool support must be address in order for the pattern-based specification to have more significant penetration in industrial settings.

Overall, the specification patterns and the ProPas tool seem to be a promising combination for a pattern-based specification of requirements in industrial settings. The method and the tool in general are easy to use and the quality of the specifications is perceived as good by the industrial practitioners. The qualitative data also suggests that the combination of specification patterns and ProPas tool seems to reduce the specification effort for the engineers. On the other hand, the main challenge for the pattern-based specification of requirements for engineers who are not skilled in formal methods is the validation of the expressed behavior. This is observed by the fact that the subjects express neutral and in some cases negative opinion (S_6 and S_7) on the quality of the generated specifications and the confidence in what they are doing.

7 RELATED WORK

In the literature, there exists a body of works that used the specification patterns for capturing industrial requirements from various domains. In the following, we list some of the recent attempts.

To the best of the authors' knowledge, there are only two reported cases in the literature that are focused on accessing the suitability of the specification patterns for specifying requirements in the automotive domain. The first case study was reported by Post et al. (Post et al., 2012). In their work, the authors conduct a case study at BOSCH, one of the leading suppliers in automotive domain, in which they aim to study whether the specification patterns are expressive enough to formalize requirements in the automotive domain. The results from the case study suggest that the specification patterns are expressive enough for capturing most of the requirements used in the study. Moreover, they also observed that the small subset of patterns are enough to formalize most of the requirements. Since the results were obtained based on a rather limited set of requirements and involved only one company, it is hard to generalize the results to the entire domain. In order to further strengthen or disprove the claims of Post et al., a follow up case study was carried out at Scania by Filipovikj et al. (Filipovikj et al., 2014). The case study involved one hundred requirements from four different systems. The results of the study are aligned with the ones of Post et al., and as such represent one step close towards being able to generalize for the whole domain. Compared to our current study, both of the previous attempts were completely oriented towards accessing the expressiveness of the specification patterns, and as such were performed by formal methods experts and not actual engineers. Also, neither of the studies involved a specialized tool support since the experts where comfortable with using general purpose text editors or pen and paper.

In order to support potential users of the specification patterns, be they from academia or industry, number of research tools of various maturity have been proposed. The Property Specifier tool (Prospec) (Mondragon and Gates, 2004) is one of the first tools proposed to facilitate the usage of specification patterns for users not trained in formal methods. It provides a complex interface that supports many degrees of freedom when specifying complex properties. Another similar tool is the Property Elucidator (PROPEL) tool (Smith et al., 2002). Both tools represent the pattern-expressed requirements as a finite state machines, however, none of them has been evaluated by industrial practitioners, thus no data on their practical usefulness is available. Another tool for assisting non-experts in formal methods is the Property Assistant (PASS) proposed by Remenska et al. (Remenska et al., 2014). In comparison to the aforementioned tools, PASS uses UML constructs to show the specified behavior captured by the patterns. The tool also provides a multi-feature interface that can be used to specify complex system properties, however, similar to the previous two tools, it has not been evaluated by industrial practitioners. Lastly, the PSPWizard tool (Autili et al., 2015) represents the latest attempt to produce a specification-pattern-based tool. Compared to the previous tools, the PSPWizard incorporates the most comprehensive catalog of specification patterns and as such can potentially be used to specify the widest pallet of different properties. The PSPWizard tool and the pattern-based specification framework proposed in (Autili et al., 2015) have successfully been applied on an industrial use case, however, the assessment was performed by researchers (presumably experts in formal methods) and not the actual practitioners (engineers).

Compared to all of the aforementioned reports, in this case study we specifically aim at accessing the practical usefulness of the specification patterns via the ProPas tool in industrial settings. As such, the purpose of our exploratory case study is to generate an initial set of empirical data about the potential usability of the specification patterns in industrial settings, which should serve a good starting point for conducting bigger case studies of similar type in order to empirically access the usefuleness of the specification patterns.

8 CONCLUSIONS

In this paper, we presented an exploratory case study through which we gather empirical data for the practical usefulness of the specification patterns via the ProPas tool for specifying system requirements in industrial settings, more precisely in the automotive domain.

The reported case study was performed in collaboration with Scania, one of the world's leading manufacturer of heavy-load vehicles, including trucks and buses. As case-study subjects, we used five stakeholders from the embedded software development process at Scania. The case-study subjects belong to the following roles: two development engineers, one requirements coach, one safety architect and one systems architect. The case study was executed in three major phases: preparation, data gathering and analysis, and reporting. In the first phase, the subjects were provided with a short tutorial on the specification patterns and the ProPas tool. In the second phase, we visited the subjects at the company premises where they performed the specification of the requirements using the ProPas tool and answered a survey, such that both quantitative and qualitative data is collected. For the quantitative data we have considered the following: correctness of the specification and time for specification, whereas for collecting the qualitative data we used a survey of ten statements, which was completed by the case-study subjects immediately after the specification of the requirements.

The analysis of the quantitative data shows the following: the overall specification correctness is 90%, meaning that only three out of thirty requirements specification instances were incorrectly specified by the subjects. The correctness results do not show a direct causal relationship between the complexity of the requirements from the system specification and the correctness of the same. The average time for formalizing the predefined set of requirements is 26.4 minutes, or 4.4 minutes per requirement. If we look at the specification time per individual requirement (see Table 3), it is clear that the complexity of the requirement has a direct effect on the specification time. For instance, the average specification time for the most complex requirement (R_6) is 12.16 minutes, which is very close to the time spend for the specification of the rest of the five requirements combined. The qualitative data, which was obtained via survey (see Section 4.1) shows that the engineers are positively inclined towards system specification via specification patterns supported by an adequate tool support, such as the ProPas tool. However, the same data reveals that in order to have more penetration and real industrial impact, the specialized tool support must be significantly improved, especially in the following aspects: provide a validation mechanism that should help non-experts in formal methods to validate the specified behavior, as well as requirements management features such as traceability, versioning, etc. Despite the identified drawbacks, our results show that the specification patterns that are supported by an adequate tooling can be practically useful for formalizing requirements by engineers and other stakeholders who are not experts in formal methods.

Based on the results from the case study, we envision the following directions for future research. First, we aim to define a set of hypotheses to be tested in future case studies, which should involve more systems, subjects and possibly include other companies, either from the same or different domain. The second direction for future work is to further improve the specialized tool support, meaning to enrich the ProPas tool with new features that address the recommendations and suggestions extracted from the qualitative data. This direction involves both engineering (tool implementation) and research efforts (finding a suitable method for validation of specified behavior).

ACKNOWLEDGEMENTS

This work has been funded by the Swedish Governmental Agency for Innovation Systems (VINNOVA) under the VeriSpec project 2013-01299.

REFERENCES

Alur, R., Courcoubetis, C., and Dill, D. (1993). Model-checking in dense real-time. *Information and computation*, 104(1):2–34.

Alur, R. and Henzinger, T. A. (1993). Real-time logics: Complexity and expressiveness. *Information and Computation*, 104(1):35–77.

Autili, M., Grunske, L., Lumpe, M., Pelliccione, P., and Tang, A. (2015). Aligning qualitative, real-time, and probabilistic property specification patterns using a structured english grammar. *IEEE Transactions on Software Engineering*, 41(7):620–638.

Barnat, J., Bauch, P., Beneš, N., Brim, L., Beran, J., and Kratochvíla, T. (2016). Analysing sanity of requirements for avionics systems. *Formal Aspects of Computing*, 28(1):45–63.

Dwyer, M. B., Avrunin, G. S., and Corbett, J. C. (1998). Property specification patterns for finite-state verification. In *Proceedings of the second workshop on Formal methods in software practice*, pages 7–15. ACM.

Filipovikj, P., Jagerfield, T., Nyberg, M., Rodriguez-Navas, G., and Seceleanu, C. (2016). Integrating pattern-based formal requirements specification in an industrial tool-chain. In *Computer Software and Applications Conference (COMPSAC), 2016 IEEE 40th Annual*, volume 2, pages 167–173. IEEE.

Filipovikj, P., Nyberg, M., and Rodriguez-Navas, G. (2014). Reassessing the pattern-based approach for formalizing requirements in the automotive domain. In *Proceedings of the 22nd IEEE International Requirements Engineering Conference (RE)*, volume 00, pages 444–450, Los Alamitos, CA, USA. IEEE Computer Society.

Filipovikj, P., Rodriguez-Navas, G., Nyberg, M., and Seceleanu, C. (2018). Automated SMT-based consistency checking of industrial critical requirements. *ACM SIGAPP Applied Computing Review*, 17(4):15–28.

Grunske, L. (2008). Specification patterns for probabilistic quality properties. In *2008 ACM/IEEE 30th International Conference on Software Engineering*, pages 31–40. IEEE.

Kitchenham, B., Pickard, L., and Pfleeger, S. L. (1995). Case studies for method and tool evaluation. *IEEE software*, 12(4):52–62.

Konrad, S. and Cheng, B. H. C. (2005). Real-time specification patterns. In *Proceedings of the 27th International Conference on Software Engineering*, ICSE '05, pages 372–381, New York, NY, USA. ACM.

Mondragon, O. A. and Gates, A. Q. (2004). Supporting elicitation and specification of software properties through patterns and composite propositions. *International Journal of Software Engineering and Knowledge Engineering*, 14(01):21–41.

Moser, L. E., Ramakrishna, Y., Kutty, G., Melliar-Smith, P. M., and Dillon, L. K. (1997). A graphical environment for the design of concurrent real-time systems. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 6(1):31–79.

Post, A., Hoenicke, J., and Podelski, A. (2011). rt-inconsistency: a new property for real-time requirements. In *International Conference on Fundamental Approaches to Software Engineering*, pages 34–49. Springer.

Post, A., Menzel, I., Hoenicke, J., and Podelski, A. (2012). Automotive behavioral requirements expressed in a specification pattern system: a case study at bosch. *Requirements Engineering*, 17(1):19–33.

Remenska, D., Willemse, T. A., Templon, J., Verstoep, K., and Bal, H. (2014). Property specification made easy: Harnessing the power of model checking in uml designs. In *International Conference on Formal Techniques for Distributed Objects, Components, and Systems*, pages 17–32. Springer.

Robson, C. and McCartan, K. (2016). *Real world research*. John Wiley & Sons.

Runeson, P. and Höst, M. (2009). Guidelines for conducting and reporting case study research in software engineering. *Empirical Softw. Engg.*, 14(2):131–164.

Smith, R. L., Avrunin, G. S., Clarke, L. A., and Osterweil, L. J. (2002). Propel: an approach supporting property elucidation. In *Proceedings of the 24th International Conference on Software Engineering*, pages 11–21. ACM.

Westman, J. and Nyberg, M. (2014). Environment-Centric Contracts for Design of Cyber-Physical Systems. In *MODELS*, pages 218–234.

Wohlin, C., Runeson, P., Höst, M., Ohlsson, M. C., Regnell, B., and Wesslén, A. (2012). *Experimentation in software engineering*. Springer Science & Business Media.